12-17-2004

A Comparison of Multiple-Model Target Tracking Algorithms

Ryan Pitre

University of New Orleans

A COMPARISON OF MULTIPLE-MODEL TARGET TRACKING ALGORITHMS

A Thesis

Submitted to the Graduate Faculty of the
University Of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
The Department of Electrical Engineering

by

Ryan Pitre

B.S.EE University of New Orleans, 2002

December 2004

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

There are many multiple-model (MM) target-tracking algorithms that are available but there has yet to be a comparison that includes all of them. This work compares seven of the currently most popular MM algorithms in terms of performance, credibility, and computational complexity. The algorithms to be considered are the autonomous multiple-model algorithm, generalized pseudo-Bayesian of first order, generalized pseudo-Bayesian of second order, interacting multiple-model algorithm, B-Best algorithm, Viterbi algorithm, and reweighted interacting multiple-model algorithm. The algorithms were compared using three scenarios consisting of maneuvers that were both in and out of the model set. Based on this comparison, there is no clear-cut best algorithm but the B-best algorithm performs best in terms of tracking errors and the IMM algorithm has the best computational complexity among the algorithms that have acceptable tracking errors.

1. INTRODUCTION

There are many target tracking algorithms available today and selecting one from these algorithms may be difficult if done so without being familiar with the availability and the quality of each algorithm. This thesis describes and compares seven of the currently most popular multiple-model (MM) algorithms used for maneuvering target tracking. The MM target-tracking algorithms to be covered are Autonomous MM (AMM), Generalized Pseudo-Bayesian of first-order (GPB1), Generalized Pseudo-Bayesian of second-order (GPB2), Interacting MM (IMM), reweighted IMM (RIMM), B-Best, and the Viterbi algorithm (VA). These MM algorithms all use multiple models simultaneously for tracking a maneuvering target. The MM approach is considered by the many as the mainstream method for tracking a maneuvering target under motion uncertainty. Based on this comparison, there is no clear-cut best algorithm but B-Best but B-Best performs best in RMSE. IMM performed best in computational complexity among the algorithms that had acceptable RMSEs.

Why Multiple-Models

It is beneficial to use more than one model in a tracking algorithm when the dynamics of the target are unknown. For example, if only one model is used and it is an incorrect model, then it is possible that the filter will yield unacceptable target state estimates or even lose the target. In order to account for the uncertainty of the target's dynamics, a multiple-model target-tracking algorithm runs a set of filters that model several possible maneuvers and non-maneuver target motion. Then, the algorithm fuses the output of those filters for an overall estimate. The multiple

model approach can detect a target maneuver when the true maneuver motion is in the model set.

MM algorithms, which are estimate-decision based, are better than the decision-estimation based algorithms because the latter first decides which model is the most probable and then estimates the target's state using only the most probable model while the former runs a set of models and then decides which model is most likely. The decision-estimation method can have poor results because it does not account for errors resulting from using the incorrect model. MM algorithms improve on the decision-estimation by estimating the target's state by using all of the models and then deciding which models are the most likely to be true among all candidates. The MM approach is much better partly because more information is available to decide which is the correct model.

Terminology

It is necessary to define the terminology and notation used in this paper because the notation in the target tracking area widely varies. A subscript k will always denote the present time and a subscript $k-1$ will always be the time that is one sampling time before the present time. A superscript (i) will represent model (i), which can be any model in the model set. A superscript (j,i) will denote a transition from model (j) to model (i).

A mode is the truth that precisely describes a target's state evolvement over time. That is, if the mode at every sample is known and the initial states are also known then every subsequent state can be accurately predicted recursively in the absence of process noise.

The term model represents a mathematical simplification for a true mode of a target. For example, it is possible for a target to accelerate at $15\,{}^{m}\!/_{s^2}$ for some finite duration of *n* samples. The mode for that particular scenario is a constant acceleration (CA) of $15\,{}^{m}\!/_{s^2}$ for that *n* samples. It is also possible for a target to be turning with a turn-rate of ${}^{5^\circ}\!/_{\text{sec}}$ then that will be the true mode. It is easy to see that by changing the turn rate or the rate of acceleration in either of those examples there can be an infinite number of possible modes. When designing a model set these variations can result in a model mismatch. A tracker rarely knows the true mode and that is why it is necessary to use a model set that contains multiple models that can model the truth. A good model set needs to include models that account for changes in velocity and direction so that the tracking algorithm will have a better chance of predicting the target's future state evolvement which will minimize the estimation error. The size of the model set is proportional to the computational complexity of the algorithm and therefore a model set should be selected that will not add unnecessary computation. The model set used here includes the following types of models: constant velocity, constant acceleration, constant deceleration, constant turn rate of a left turn, and constant turn rate of a right turn.

Another term needed to be defined is sequence or sequence history. A mode sequence is the sequence of modes that the target was in during the last *n* samples where *n* can be any length of the history to be recorded. If the target moves at constant velocity (CV) for *n* time steps and then accelerates with a constant acceleration (CA) for another *m* time steps then that sequence will be CV for *n* time

steps followed by CA for *m* time steps. The notation, *(j,i)*, will be used when describing a mode sequence of where mode *(j)* is true at time *k*-1 and there is a transition at time *k* to mode *(i)*.

Transition probability matrix (TPM) is a matrix whose element $\pi_{j,i}$ is the probability of transition from mode (*j*) to mode (*i*). The TPM has the form

$$\pi = \begin{bmatrix} \pi_{1,1} & \pi_{2,1} & \cdots & \pi_{M,1} \\ \pi_{1,2} & \pi_{2,2} & \cdots & \pi_{M,2\cdots} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{1,M} & \pi_{2,M} & \cdots & \pi_{M,M} \end{bmatrix}$$

where *M* is the number of models in the model set. Using the total probability theorem, the sum of the rows is always equal to unity because the probability of a model transitioning to another model, represented by $\pi_{j,i}$, plus the probability of no transition, or $\pi_{j,j}$, must be equal to one. A transition probability with the sequence *(j,i)* is expressed as $\pi_{j,i}$, where *j* is the row number and *i* is the column number. The letter *j* represents the mode at *k-1* and *i* will represent the mode at time *k*. The TPM is useful when modeling the so-called Markov-jump sequence, which is necessary for the second-generation MM target-tracking algorithms. A system is a Markov jump system if the mode sequence is a Markov chain, which for a discrete-time system, can be represented by

$$P\left\{m_k^i \mid m_{k-1}^j\right\} = \pi_{j,i} \quad \forall i, j, k$$

where m_k^j represents mode *(j)* at time *k*.

The process noise covariance, *Q*, is a model parameter that indicates how much the filter trusts the model. A very small number for *Q*, such as 0.0001

represents a high level of faith in the model. A *Q* of zero means that the model is a perfect match to the truth. However, the error is likely to diverge because it is impossible to have a perfect match when there is uncertainty in target motion. A very high *Q* means that there is little faith in the model.

Tangential acceleration, a_t, is an acceleration that is in the same direction as the velocity. A normal acceleration, a_n, is an acceleration that is orthogonal to the velocity. The heading angle, ϕ, is the angle that is between the x-axis and a_t. (See figure 1) [1]. These accelerations are used in generating the maneuvers.

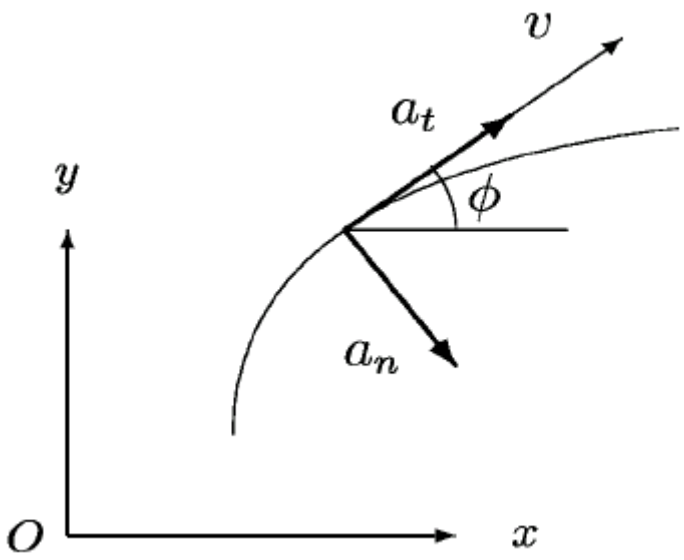

Figure 1

The remaining sections are sections: 2 MULTIPLE-MODEL ALGORITHMS, 3 IMPLEMENTING MULTIPLE-MODEL ALGORITHMS, 4 TEST SCENARIOS, 5 METHODS OF COMPARISON, 6 RESULTS, 7 DISCUSSIONS, and 8 CONCLUSIONS.

2. MULTIPLE-MODEL ALGORITHMS

The Autonomous MM algorithm, or AMM, is a first generation algorithm MM in that its models do not use information from the other models in the model set. This means that each filter in the model set works independently from all of the other models. The remaining algorithms being considered for performance comparison are second-generation MM algorithms. These algorithms have a model set in which the filters interact with each other to provide a better estimate through teamwork. This means that each filter in the set uses the estimates from the other filters to reinitialize itself and thus each filter has more information to make a better estimate.

The differences between the first and second generation MM tracking algorithms is due to their fundamental assumptions. A first generation algorithm assumes that

1. The mode does not change
2. The true mode is in the model set

The fundamental assumptions of a second-generation algorithm are

1. The true mode sequence is a Markov or semi-Markov chain
2. The true mode is in the model set

Because the second-generation algorithms allow for a change in mode, some algorithms use some combination of filter outputs for reinitialization. This reinitialization means that the state prediction, shown next for convenience,

$$\hat{x}_{k|k-1}^{(i)} = F_{k-1}^{(i)} \hat{x}_{k-1|k-1}^{(i)} + G_{k-1}^{(i)} u_{k-1}^{(i)} + \Gamma_{k-1}^{(i)} \bar{w}_{k-1}^{(i)},$$

will not use the updated estimate, $\hat{x}_{k|k}^{(i)}$, but instead will use the following equation

$$\hat{x}_{k|k-1}^{(i)} = F_{k-1}^{(i)}\bar{x}_{k-1|k-1}^{(i)} + G_{k-1}^{(i)}u_{k-1}^{(i)} + \Gamma_{k-1}^{(i)}\bar{w}_{k-1}^{(i)},$$

where the reinitialization, $\bar{x}_{k-1|k-1}^{(i)}$, is some combination of the updated estimates from the other models in the model set. The above equations will be explained in section 3 IMPLEMENTING MM TRACKING ALGORITHMS.

AMM

The autonomous MM target-tracking algorithm (AMM) is the simplest to implement of all of the algorithms to be discussed here. This method runs a Kalman filter for each model in the model set and after each of the models is updated independently, all of the updates are mixed for an overall estimate. The overall output of the tracking algorithm is the sum of the outputs of all of the individual filters weighted by their model probabilities. The model probabilities are calculated using that model's likelihood, $L_k^{(i)}$. The model likelihood for model (*i*) is calculated using the measurement prediction error, $\tilde{z}_k^{(i)}$, and the measurement prediction error covariance, $S_k^{(i)}$. Basically, $L_k^{(i)}$ measures how likely the measurement prediction error could result from model (*i*) being true at time *k* given the data at time *k-1*.

Model Likelihood**:**

$$L_k^{(i)} \triangleq p\left[\tilde{z}_k^{(i)} \mid m_{(i)}^k, z^{k-1}\right] = \frac{\exp\left[-\frac{1}{2}\left(\tilde{z}_k^{(i)}\right)'\left(S_k^{(i)}\right)^{-1}\tilde{z}_k^{(i)}\right]}{\left|2\pi S_k^{(i)}\right|^{1/2}}$$

A mode probability is the weight assigned to the output of each filter inside the algorithm. This weight provides an indication of how probable the model is in effect at time *k* compared to the other models in the model set. It is calculated as follows

Mode probability**:**

$$\mu_k^{(i)} = \frac{\mu_{k-1}^{(i)}L_k^{(i)}}{\sum_{j\in M}\mu_{k-1}^{(j)}L_k^{(j)}}$$

The mode probability is normalized so that $\sum_{i \in M} \mu_k^{(i)} = 1$.

This should make sense intuitively because the sum of all of the probabilities must be equal to unity. The overall output of the AMM algorithm fuses all of the updated states from the model set using the mode probabilities and the total probability theorem. One cycle of the AMM algorithm is shown in table 1.

One cycle of the AMM algorithm is as follows:

---

1. Model-conditioned filtering (for i =1,2,. . .,M):

$$\left(\hat{x}_{k-1|k-1}^{(i)}, P_{k-1|k-1}^{(i)}\right) \rightarrow \left(\hat{x}_{k|k}^{(i)}, P_{k|k}^{(i)}, \tilde{z}_k^{(i)} S_k^{(i)}\right)$$

2. Mode probability update (for i =1,2,. . .,M):

$$\left(\mu_{k-1}^{(i)}, \tilde{z}_k^{(i)}, S_k^{(i)}\right) \rightarrow \left(\mu_k^{(i)}, L_k^{(i)}\right)$$

3. Estimate fusion:

$$\left(\hat{x}_{k|k}^{(i)}, P_{k|k}^{(i)}, \mu_k^{(i)}\right) \rightarrow \left(\hat{x}_{k|k}, P_{k|k}\right)$$

Table 1

IMM

The most popular MM algorithm in the target tracking area is the interacting multiple-model (IMM) algorithm. One cycle of the IMM algorithm is shown in table 2. It reinitializes each model with a weighted sum of the updated estimates from every model in the model set. The weights are calculated based on two probabilities. The first being the probability that mode *(j)* was true at time *k-1* and the second is the probability that mode *(i)* will be true at time *k*. This type of reinitialization that combines similar sequences is called *merging*. Merging reduces the amount of

computation by combining similar sequences. The computational complexity of IMM is proportional to M, where M is the number of models used in the algorithm.

One cycle of the IMM algorithm is as follows:

---

1. Model-conditioned reinitialization (for i =1,2,. . .,M):

   Predicted mode probability: $\mu_{k|k-1}^{(i)} \triangleq P\left\{m_k^{(i)} \mid z^{k-1}\right\} = \sum_{j \in M} \pi_{ji} \mu_{k-1}^{(j)}$

   Mixing weight: $\mu_{k-1}^{j|i} \triangleq P\left\{m_{k-1}^{(j)} \mid m_k^{(i)}, z^{k-1}\right\} = \dfrac{\pi_{ji} \mu_{k-1}^{(j)}}{\mu_{k|k-1}^{(i)}}$

   Mixing estimate: $\bar{x}_{k-1|k-1}^{(i)} \triangleq E\left[x_{k-1} \mid m_k^{(i)}, z^{k-1}\right] = \sum_{j \in M} \hat{x}_{k-1|k-1}^{(j)} \mu_{k-1}^{j|i}$

   Mixing covariance: $\bar{P}_{k-1|k-1}^{(i)} = \sum_{j \in M} \left[P_{k-1|k-1}^{(j)} + \left(\bar{x}_{k-1|k-1}^{(i)} - \hat{x}_{k-1|k-1}^{(j)}\right)\left(\bar{x}_{k-1|k-1}^{(i)} - \hat{x}_{k-1|k-1}^{(j)}\right)^T \right] \mu_{k-1}^{j|i}$

2. Model-conditioned filtering (for i =1,2,. . .,M):

   $\left(\bar{x}_{k-1|k-1}^{(i)}, \bar{P}_{k-1|k-1}^{(i)}\right) \rightarrow \left(\hat{x}_{k|k}^{(i)}, P_{k|k}^{(i)}, \tilde{z}_k^{(i)} S_k^{(i)}\right)$

3. Mode probability update

   $\left(\mu_{k|k-1}^{(i)}, \tilde{z}_k^{(i)}, S_k^{(i)}\right) \rightarrow \left(\mu_k^{(i)}, L_k^{(i)}\right)$

4. Estimate fusion:

   $\left(\hat{x}_{k|k}^{(i)}, P_{k|k}^{(i)}, \mu_k^{(i)}\right) \rightarrow \left(\hat{x}_{k|k}, P_{k|k}\right)$

Table 2

GPB1 And GPB2

The generalized pseudo Bayesian algorithm of order n (GPBn) reinitializes each filter with the updates of sequences weighted by their sequence probabilities. A sequence history is the sequence of modes that the target was in during the last *n* time samples. GPB can be implemented using sequence histories of various lengths. The length *n* in GPBn denotes the time steps in sequence history. For example, GPB2

uses data from time *k* and time *k-1*, while GPB1 uses data from only time *k*. GPBn *merges* similar sequence histories to reinitialize each model. For example, GPB2, which uses data beginning at time *k-1*, will merge the sequence histories that are the similar up to time *k-1*. Merging for GPB2 can be summarized as follows

$$\hat{x}_{k-1|k-1}^{\left(j^{k-3},j,i\right)} \approx \hat{x}_{k-1|k-1}^{(j,i)}.$$

The computational complexity of GPBn is proportional to M^n, where M is the number of models used in the algorithm and n is the length of the sequence histories that will be used. The primary difference between IMM and GPB2 is that IMM merges the estimates before conditional filtering and GPB2 merges the estimates after the conditional filtering.

One cycle of GPB1 is shown in table 3.

---

1. Model-conditioned filtering (for i =1,2,. . .,M):
   $\left(\hat{x}_{k-1|k-1}, P_{k-1|k-1}\right) \rightarrow \left(\hat{x}_{k|k}^{(i)}, P_{k|k}^{(i)}, \tilde{z}_k^{(i)} S_k^{(i)}\right)$
2. Mode probability update (for i =1,2,. . .,M):
   $\left(\pi_{j,i}, \mu_{k-1}^{(i)}, \tilde{z}_k^{(i)}, S_k^{(i)}\right) \rightarrow \left(L_k^{(i)}, \mu_k^{(i)}\right)$
3. Estimate fusion:
   $\left(\hat{x}_{k|k}^{(i)}, P_{k|k}^{(i)}, \mu_k^{(i)}\right) \rightarrow \left(\hat{x}_{k|k}, P_{k|k}\right)$

Table 3

One cycle of GPB2 is shown in table 4

1. Model-conditioned filtering (for every transition (j,i)):

$$\left(\bar{x}_{k-1|k-1}^{(j)}, \bar{P}_{k-1|k-1}^{(j)}\right) \rightarrow \left(\hat{x}_{k|k}^{(j,i)}, P_{k|k}^{(j,i)}, \tilde{z}_k^{(j,i)}, S_k^{(j,i)}\right)$$

2. Model-conditioned reinitialization (for i =1,2,. . .,M):

Predicted mode probability: $\mu_{k|k-1}^{(i)} \triangleq P\left\{m_k^{(i)} \mid z^{k-1}\right\} = \sum_{j \in M} \pi_{ji} \mu_{k-1}^{(j)}$

Sequence mixing weight: $\mu_k^{j|i} \triangleq P\left\{m_{k-1}^{(j)} \mid m_k^{(i)}, z^{k-1}\right\} = \dfrac{\pi_{ji} \mu_{k-1}^{(j)}}{\mu_{k|k-1}^{(i)}}$

Mixing estimate: $\bar{x}_{k|k}^{(i)} \triangleq E\left[x_k \mid m_k^{(i)}, z^k\right] = \sum_{j \in M} \hat{x}_{k|k}^{(j,i)} \mu_k^{j|i}$

Mixing covariance: $\bar{P}_{k|k}^{(i)} = \sum_{j \in M} \left[P_{k|k}^{(j,i)} + \left(\bar{x}_{k|k}^{(i)} - \hat{x}_{k|k}^{(j,i)}\right)\left(\bar{x}_{k|k}^{(i)} - \hat{x}_{k|k}^{(j,i)}\right)^T\right] \mu_k^{j|i}$

3. Mode probability update

Predicted mode probability:

$$\left(\mu_{k|k-1}^{(j,i)}, \tilde{z}_k^{(j,i)}, S_k^{(j,i)}\right) \rightarrow \left(L_k^{(j,i)}, \mu_k^{(j,i)}\right)$$

3. Estimate fusion:

$$\left(\hat{x}_{k|k}^{(j,i)}, P_{k|k}^{(j,i)}, \mu_k^{(j,i)}\right) \rightarrow \left(\hat{x}_{k|k}, P_{k|k}\right)$$

Table 4

B-Best

B-Best algorithm also uses the pruning method to reduce the number of mode sequences histories. It runs each of its M filters B times and keeps only the B-best sequences with the highest probabilities of all $B \times M$ sequences. These B sequences are used for the next iteration. One cycle of the B-Best MM algorithm is shown in table 5. The set containing the B-best sequences is denoted as β and j will be the

terminating mode at *k-1* and the initialization mode for *k*. B-Best algorithm has a computational complexity proportional to $B \times M$.

One cycle of the B-Best algorithm is as follows:

---

1. Model-conditioned filtering (for every transition (j,i) *where* $j \in \beta_{k-1}$):
   $\left(\bar{x}_{k-1|k-1}^{(j)}, \bar{P}_{k-1|k-1}^{(j)}\right) \rightarrow \left(\hat{x}_{k|k}^{(j,i)}, P_{k|k}^{(j,i)}, \tilde{z}_k^{(j,i)} S_k^{(j,i)}\right)$
2. Sequence probability update (for every transition (j,i) *where* $j \in \beta_{k-1}$):
   $\left(\pi_{j,i}, \mu_{k-1}^{(j)}, \tilde{z}_k^{(j,i)}, S_k^{(j,i)}\right) \rightarrow \left(L_k^{(j,i)}, \mu_k^{(j,i)}\right)$
3. Update the 'B' best sequences for time *k*
   Most probable sequences: β_k=B sequences with maximum $\mu_k^{(j,i)}$*:*
4. Sequence reinitialization $\forall (j,i) \in \beta_k$
   $\bar{x}_{k|k}^{(i)} = \hat{x}_{k|k}^{(j,i)}$
   $\bar{P}_{k|k}^{(i)} = P_{k|k}^{(j,i)}$
   $\mu_k^{(i)} = \dfrac{\mu^{(j,i)}}{\sum_{(j,i)\in\beta} \mu^{(j,i)}}$
5. Estimate fusion (for every transition (j,i) *where* $j \in \beta_k$):
   $\left(\bar{x}_{k|k}^{(i)}, \mu_k^{(i)}, \bar{P}_{k|k}^{(i)}\right) \rightarrow \left(\hat{x}_{k|k}, P_{k|k}\right)$

Table 5

Viterbi Algorithm

Viterbi algorithm (VA) finds the best sequence history for every model in the model set. It guarantees to have the most likely sequence in the set of sequence histories. This algorithm uses what is called the *pruning* method. Pruning reduces computational complexity by removing sequence histories that are unlikely to be true given the data. Basically, given the model probabilities for each model in the model set at time *k-1* and also given the transition probabilities, each model is reinitialized

using the updated estimate from the model with the highest transition. The VA used here fuses the estimates using a weighted sum. Another implementation of the VA uses the most likely sequence as the overall estimate. One cycle of the Viterbi algorithm is shown in table 6.

One cycle of the Viterbi algorithm is as follows:

---

1. Model-conditioned filtering (for every transition (j,i)):
   $\left(\overline{x}_{k-1|k-1}^{(j)}, \overline{P}_{k-1|k-1}^{(j,)}\right) \rightarrow \left(\hat{x}_{k|k}^{(j,i)}, P_{k|k}^{(j,i)}, \tilde{z}_k^{(j,i)} S_k^{(j,i)}\right)$
2. Mode probability update (for every transition (j,i)):
   $\left(\pi_{j,i}, \mu_{k-1}^{(j,i)}, \tilde{z}_k^{(j,i)}, S_k^{(j,i)}\right) \rightarrow \left(\mu_k^{(j,i)}, L_k^{(j,i)}\right)$
3. Model-conditioned reinitialization (for i =1,2, . . .,M):

   Most likely (j): $(j) = \arg\max_{(j)} \mu_k^{(j,i)}$

   State reinitialization: $\overline{x}_{k|k}^{(i)} = \hat{x}_{k|k}^{(j,i)}$

   Covariance reinitialization: $\overline{P}_{k|k}^{(i)} = \hat{P}_{k|k}^{(j,i)}$
4. Estimate fusion: $\left(\overline{x}_{k|k}^{(i)}, \overline{P}_{k|k}^{(i)}, \mu_k^{(i)}\right) \rightarrow \left(\hat{x}_{k|k}, P_{k|k}\right)$

Table 6

RIMM

Reweighted IMM or RIMM is an implementation of expectation maximization (EM) which maximizes the likelihood function by treating the mode sequence as missing data. RIMM is similar to IMM except for the weights used to reinitialize each filter and the formulas used to calculate the output. RIMM requires M^2 predictions and M updates while IMM requires only M predictions and M updates. One cycle of the RIMM target-tracking algorithm is shown in table 7.

One cycle of the RIMM algorithm is as follows:

---

1. Model-conditioned prediction probabilities (for i =1,2,. . .,M):

   Predicted mode probability: $\mu_{k|k-1}^{(i)} \triangleq P\left\{m_k^{(i)} \mid z^{k-1}\right\} = \sum_{j\in M} \pi_{ji}\mu_{k-1}^{(j)}$

   Mixing weight: $\mu_{k-1}^{j|i} \triangleq P\left\{m_{k-1}^{(j)} \mid m_k^{(i)}, z^{k-1}\right\} = \dfrac{\pi_{ji}\mu_{k-1}^{(j)}}{\mu_{k|k-1}^{(i)}}$

2. Model-conditioned prediction (for every transition (j,i)):

   Mixing covariance: $P_{k|k-1}^{(j,i)} = \dfrac{\pi_{ji}}{\mu_{k|k-1}^{(i)}} F_k^{(i)} P_{k-1|k-1}^{(j)} \left(F_k^{(i)}\right)^T + G_k^{(i)} Q_k^{(i)} \left(G_k^{(i)}\right)^T$

   Mixing estimate: $\hat{x}_{k|k-1}^{(j,i)} = E\left[x_k \mid z^{k-1}, m_{k-1}^{(j)}, m_k^{(i)}\right] = F_k^{(i)} \hat{x}_{k-1|k-1}^{(j)} \left(F_k^{(i)}\right)^T + G_k^{(i)} u_{k-1}^{(i)}$

   Predicted state: $\hat{x}_{k|k-1}^{(i)} = P_{k|k-1}^{(i)} \sum_{j\in M} \left(P_{k|k-1}^{(j,i)}\right)^{-1} \hat{x}_{k|k-1}^{(j,i)} \mu_{k-1}^{(j|i)}$

   Predicted covariance: $\left(P_{k|k-1}^{(i)}\right)^{-1} = \sum_{j\in M} \left(P_{k|k-1}^{(j,i)}\right)^{-1} \mu_{k-1}^{(j|i)}$

3. Model-conditioned filtering (for i =1,2,. . .,M):

   $\left(\hat{x}_{k|k-1}^{(i)}, P_{k|k-1}^{(i)}\right) \rightarrow \left(\hat{x}_{k|k}^{(i)}, P_{k|k}^{(i)}, \tilde{z}_k^{(i)}, S_k^{(i)}\right)$

4. Mode probability update

   $\left(\mu_{k|k-1}^{(i)}, \tilde{z}_k^{(i)}, S_k^{(i)}\right) \rightarrow \left(\mu_k^{(i)}, L_k^{(i)}\right)$

5. Estimate fusion:

   Overall covariance: $P_{k|k}^{-1} = \sum_{i\in M} \left(P_{k|k}^{(i)}\right)^{-1} \mu_k^{(i)}$

   Overall estimate: $\hat{x}_{k|k} = P_{k|k} \sum_{i\in M} \left(P_{k|k}^{(i)}\right)^{-1} \hat{x}_{k|k}^{(i)} \mu_k^{(i)}$

Table 7

For more information about these algorithms the reader is referred to [2].

3. IMPLEMENTING MM TARGET TRACKING ALGORITHMS

This section explains the steps taken to design each of the MM algorithms and includes a description of the Kalman filter, the model set design, and filter initializations.

The Kalman Filter

The Kalman filter is used to estimate the state for each model and therefore it is essential to have a basic understanding of the Kalman filter in order to understand the MM algorithms. One fundamental assumption of the Kalman filter is that the measurement noise and the process noise are Gaussian and uncorrelated. The Kalman filter's eight equations can be divided up into two parts, prediction and update. The prediction part consists of the first five equations and the update part consists of equations six through eight. The steps of the Kalman filter are as follows:

Prediction:

$$\hat{x}_{k|k-1} = F_{k-1}\hat{x}_{k-1|k-1} + G_{k-1}u_{k-1} + \Gamma_{k-1}\bar{w}_{k-1} \quad \textbf{(1)}$$

$$\hat{z}_{k|k-1} = H_k\hat{x}_{k|k-1} + \bar{v}_k \quad \textbf{(2)}$$

$$P_{k|k-1} = F_{k-1}P_{k-1|k-1}F_{k-1}^T + \Gamma_{k-1}Q_{k-1}\Gamma_{k-1}^T \quad \textbf{(3)}$$

$$S_k = H_kP_{k|k-1}H_{k-1}^T + R_k \quad \textbf{(4)}$$

$$K_k = P_{k|k-1}H_k^TS_k^{-1} \quad \textbf{(5)}$$

Update:

$$\tilde{z}_k = z_k - \hat{z}_{k|k-1} \quad \textbf{(6)}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\tilde{z}_k \quad \textbf{(7)}$$

$$P_{k|k} = P_{k|k-1} - K_kS_kK_k^T \quad \textbf{(8)}$$

The first step,

$$\hat{x}_{k|k-1} = F_{k-1}\hat{x}_{k-1|k-1} + G_{k-1}u_{k-1} + \Gamma_{k-1}\overline{w}_{k-1}$$

predicts the state x_k given the update $\hat{x}_{k-1|k-1}$, which is calculated in step (7) during the previous iteration. This predicted state is called state prediction. The F_k term, which models the systems dynamics, is the state-transition matrix at time *k* meaning that in the ideal case without process noise, one can determine the next state of each of the parameters in *x* by using the relationship $x_k = F_k x_{k-1}$. Process noise is modeled with the 'w' term and process input is modeled in the *u* term. The *G* and the Γ terms are matrices that represent the mathematical relationship between the noise and input parameters to the state parameters. More will be said on the *F, G,* and Γ matrices, which make up the system dynamics, in the ***Model Set Design*** section.

The second step,

$$\hat{z}_{k|k-1} = H_k\hat{x}_{k|k-1} + \overline{v}_k$$

is the measurement prediction. It attempts to predict what the measurement will be at time *k* based on $\hat{x}_{k|k-1}$ and the mean of the measurement noise and the input. The H_k term represents the relationship between the state and the measurement.

The third step,

$$P_{k|k-1} = F_{k-1}P_{k-1|k-1}F_{k-1}^T + \Gamma_{k-1}Q_{k-1}\Gamma_{k-1}^T$$

calculates the predicted covariance of the estimate using the state-transition matrix, F_k, and the H_k matrix. The predicted covariance is an indicator of how poor an

estimate is expected to be. A covariance equal to zero indicates perfect knowledge while a large covariance indicates poor estimation/prediction accuracy.

The fourth step,

$$S_k = H_k P_{k|k-1} H_{k-1}^T + R_k$$

calculates the measurement prediction covariance or $S_k = \mathrm{cov}(\tilde{z}_k)$. This is useful because it is an indicator of the quality of the measurement prediction. The fifth step,

$$K_k = P_{k|k-1} H_k^T S_k^{-1}$$

calculates the filter gain that is used to weight the measurement prediction error in equation (7). This gain is proportional to the prediction covariance and inversely proportional to the measurement prediction covariance. If the measurement prediction covariance is very small or the prediction covariance is large, meaning that the estimate may not be very good, more weight is given to the measurement and the filter gain will be large. When the prediction covariance is small or the measurement prediction covariance is large then the filter gain will be small which means that the prediction is more reliable.

Step six, $\tilde{z}_k = z_k - \hat{z}_{k|k-1}$, simply calculates the difference between the measurement and measurement prediction. This difference is called the measurement prediction error, or the residual. Step seven,

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{z}_k ,$$

uses the filter gain to update the state estimate. Step eight,

$$P_{k|k} = P_{k|k-1} - K_k S_k K_k^T ,$$

calculates the updated covariance. The estimates, $\hat{x}_{k|k}$ and $P_{k|k}$, are the outputs of the Kalman filter at each iteration.

Model Set Design

Each MM target-tracking algorithm uses the following nine target motion models:

Constant velocity

- Constant velocity (CV)

Constant acceleration

- Constant acceleration of $67\ {}^{m}/_{\text{sec}^2}$ $\text{CA}\left(67\ {}^{m}/_{\text{s}^2}\right)$
- Constant acceleration of $33\ {}^{m}/_{\text{sec}^2}$ $\text{CA}\left(33\ {}^{m}/_{\text{s}^2}\right)$
- Constant acceleration of $-67\ {}^{m}/_{\text{sec}^2}$ $\text{CA}\left(-67\ {}^{m}/_{\text{s}^2}\right)$
- Constant acceleration of $-33\ {}^{m}/_{\text{sec}^2}$ $\text{CA}\left(-33\ {}^{m}/_{\text{s}^2}\right)$

Constant turn rate

- Constant left turn with 14° per second turn rate $\text{CT}\left(14^{\circ}/_{s}\right)$
- Constant left turn with 7° per second turn rate $\text{CT}\left(7^{\circ}/_{s}\right)$
- Constant right turn with -14° per second turn rate $\text{CT}\left(-14^{\circ}/_{s}\right)$
- Constant right turn with -7° per second turn rate $\text{CT}\left(-7^{\circ}/_{s}\right)$

These nine models cover the four different target maneuvering cases and the case with no maneuver. Originally, only five models were used to cover the four maneuvering cases and the one non-maneuvering case. The original model set

appeared to be too spread out so a model set was adopted that covered a better range of accelerations. The new model set consists of nine models that cover a wider range of maneuvers and therefore is able to be closer to the true mode.

Constant velocity

The CV model assumes a nearly constant velocity with

$$F = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad u_k = 0$$

in this comparison, $T = 1\,\text{sec}$. This is the most basic of models because it assumes no acceleration. This F matrix is the state-transition matrix and is given above. The state prediction equation is

$$x_k = Fx_{k-1} + Gu_{k-1}$$

where

$$x_k = \begin{bmatrix} x_{k-1} \\ \dot{x}_{k-1} \\ y_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} \qquad G = \begin{bmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{bmatrix}$$

This is the standard format used for all of the models in the model set.

Constant acceleration models

The CA models used assume a preset, tangential acceleration that is in the same direction as the velocity. The direction of the velocity is calculated each iteration by determining the unit vector, $|v|$, in the direction of the sum of the *x* and *y* velocity components. See Figure 2.

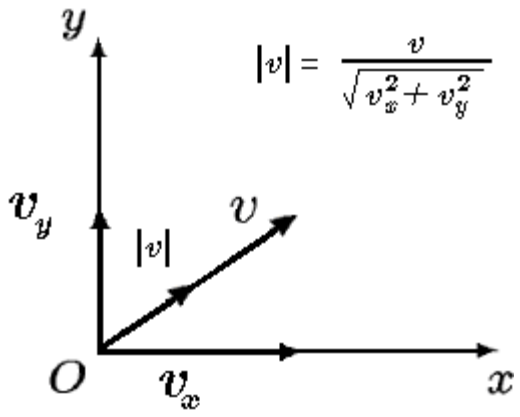

Figure 2

Once the unit vector components are obtained, the preset tangential acceleration is multiplied by the *x* and y unit-vector velocity components to get the acceleration components. This separation of the acceleration into *x* and *y* components is done for all of the acceleration models.

The larger tangential accelerations were selected to be just less than 7g, which is the maximum possible acceleration in the test scenarios. The smaller accelerations, $33\ m/s^2$ and $-33\ m/s^2$, were selected to cover the accelerations in between the maximum acceleration model and the CV model. These acceleration values allow the model set to cover the range of possible accelerations for this problem. In addition, the values selected for constant acceleration gives enough separation between the acceleration models so that the filter will be able to distinguish the acceleration models from each other and the CV model. The state transition matrix and the acceleration input for the models are:

$$F = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad , \qquad u_k = a_T^{(i)} \cdot \begin{bmatrix} \dfrac{\dot{x}_{k-1}}{\sqrt{\dot{x}_{k-1}^2 + \dot{y}^2}_{k-1}} \\ \dfrac{\dot{y}_{k-1}}{\sqrt{\dot{x}_{k-1}^2 + \dot{y}_{k-1}^2}} \end{bmatrix} m/s^2$$ where $\dot{x}$ and $\dot{y}$ are the velocity components and $a_T^{(i)}$ is the tangential acceleration for model *(i)*.

Constant turn models

The CT models assume a nearly constant turn to the either the right or the left. The values for the turn rate, ω, were selected because a turn rate of $\omega = 14$ degrees/sec is approximately a 7g turn when the target has a velocity of 330 m/s, which is approximately the speed of sound. A turn rate of $\omega = 7$ degrees/sec was selected because it is about half of 7g under the previous speed assumption. The *F* matrix and the input for the constant turn models is

$$F = \begin{bmatrix} 1 & \dfrac{\sin\omega T}{\omega} & 0 & -\dfrac{1-\cos\omega T}{\omega} \\ 0 & \cos\omega T & 0 & -\sin\omega T \\ 0 & \dfrac{1-\cos\omega T}{\omega} & 1 & \dfrac{\sin\omega T}{\omega} \\ 0 & \sin\omega T & 0 & \cos\omega T \end{bmatrix} \qquad u_k = 0$$

Filter Initializations

The TPM used for this comparison is not time varying and its elements were initialized as follows with the rows as follows: CV, CT$\left(14^\circ/s\right)$, CT$\left(-14^\circ/s\right)$, CT$\left(7^\circ/s\right)$, CT$\left(-7^\circ/s\right)$,CA$\left(67\ m/s^2\right)$,CA$\left(-67\ m/s^2\right)$, CA$\left(33\ m/s^2\right)$, and CA$\left(-33\ m/s^2\right)$

$$\pi = \begin{bmatrix} 0.99 & 0.00125 & 0.00125 & 0.00125 & 0.00125 & 0.00125 & 0.00125 & 0.00125 & 0.00125 \\ 0.193 & 0.75 & 0.001 & 0.05 & 0.001 & 0.00125 & 0.00125 & 0.00125 & 0.00125 \\ 0.193 & 0.001 & 0.75 & 0.001 & 0.05 & 0.00125 & 0.00125 & 0.00125 & 0.00125 \\ 0.00369 & 0.00125 & 0.00001 & 0.99 & 0.00005 & 0.00125 & 0.00125 & 0.00125 & 0.00125 \\ 0.00369 & 0.00001 & 0.00125 & 0.00005 & 0.99 & 0.00125 & 0.00125 & 0.00125 & 0.00125 \\ 0.193 & 0.00125 & 0.00125 & 0.00125 & 0.00125 & 0.75 & 0.001 & 0.05 & 0.001 \\ 0.193 & 0.00125 & 0.00125 & 0.00125 & 0.00125 & 0.001 & 0.75 & 0.001 & 0.05 \\ 0.00369 & 0.00125 & 0.00125 & 0.00125 & 0.00125 & 0.00125 & 0.00001 & 0.99 & 0.00005 \\ 0.00369 & 0.00125 & 0.00125 & 0.00125 & 0.00125 & 0.00001 & 0.00125 & 0.00005 & 0.99 \end{bmatrix}$$

The method used to initialize $\hat{x}_0$ and P_0 for the Kalman filter is the weighted least-squares (WLS) method because this initialization has been shown to be better than two-point differencing (TPD) in that the filter converges to its steady state faster [1]. The only difference between TPD and WLS for CV and CA models is that WLS adds an additional term to P_0.

$$P_0^{WLS} = P_0^{TPD} + \begin{bmatrix} 0 & 0 \\ 0 & T^2 \sigma_w^2 / 4 \end{bmatrix}, \quad P_0^{TPD} = \sigma_v^2 \begin{bmatrix} 1 & 1/T \\ 1/T & 2/T^2 \end{bmatrix}$$

where σ_w^2 is the variance of the process noise and σ_v^2 is the variance of the measurement noise. TPD and WLS have the same initialization for the state, which is [1]

$$\hat{\mathbf{x}}_0 = \begin{bmatrix} \hat{x} \\ \dot{\hat{x}} \end{bmatrix} = \begin{bmatrix} z_0 \\ \dfrac{z_0 - z_{-1}}{T} \end{bmatrix}.$$

For CT models, the WLS initialization is [1]

$$\hat{\mathbf{x}}_0 = \begin{bmatrix} \hat{x} \\ \dot{\hat{x}} \\ \hat{y} \\ \dot{\hat{y}} \end{bmatrix} = \begin{bmatrix} z_{0,x} \\ \dfrac{\omega\left[\sin(\omega T)\left(z_{0,x} - z_{-1,x}\right) - \left(1-\cos(\omega T)\right)\left(z_{0,y} - z_{-1,y}\right)\right]}{2\left(1-\cos(\omega T)\right)} \\ z_{0,y} \\ \dfrac{\omega\left[\sin(\omega T)\left(z_{0,y} - z_{-1,y}\right) - \left(1-\cos(\omega T)\right)\left(z_{0,x} - z_{-1,x}\right)\right]}{2\left(1-\cos(\omega T)\right)} \end{bmatrix}$$

$$P_0^{WLS} = \begin{bmatrix} \sigma_v^2 & c & 0 & \dfrac{\omega\sigma_v^2}{2} \\ c & p_{22} & -\dfrac{\omega\sigma_v^2}{2} & 0 \\ 0 & -\dfrac{\omega\sigma_v^2}{2} & \sigma_v^2 & c \\ \dfrac{\omega\sigma_v^2}{2} & 0 & c & p_{44} \end{bmatrix}$$

where $c = \dfrac{\omega\sigma_v^2 \sin(\omega T)}{2\left(1-\cos(\omega T)\right)}$ and

$p_{22} = p_{44} = \dfrac{8\sigma_v^2\omega^2 + \sigma_w^2\omega^2 T^4 - 4\sigma_w^2 T^3 \omega \sin(\omega T) + 8\sigma_w^2 T^2\left(1-\cos(\omega T)\right)}{8\left(1-\cos(\omega T)\right)}$. Unfortunately, the WLS initialization for the CT model was not used because it was overlooked and two-point differencing was used instead.

The probability of the CV model was initialized to 0.99 and the remaining model probabilities were all initialized to the same value, 0.00125

The last parameter to be initialized is *Q*. Each algorithm was initialized with the same *Q*. The *Q* has 5 m^2 for the CV model and 2500 m^2 for the maneuver model. These values provide an acceptable RMSE during steady state with a tradeoff in peak error and quick transition between states based on experimentation with other values of Q.

4. TEST SCENARIOS

Three scenarios were used to compare the seven MM tracking algorithms. The ground truth trajectories were generated using the following discretized 2D curvilinear motion-model [3]:

$$x_{k+1} = x_k + Tv_k \cos\phi_k + w_{x_k}$$

$$y_{k+1} = y_k + Tv_k \sin\phi_k + w_{y_k}$$

$$v_{k+1} = v_k + Ta_{t_k} + w_{v_k}$$

$$\phi_{k+1} = \phi_k + T\,{a_{n_k}}/{V_k} + w_{\phi_k}$$

Where x, y, v, ϕ are the target positions, velocity, and heading, respectively. See figure 1. For each Monte Carlo run $i = 1,2,3,\ldots,N$ where $N = 100$

$$x_0^{(i)} \sim N\left(\bar{x}_0, \sigma_{x_0}^2\right), \quad y_0^{(i)} \sim N\left(\bar{y}_0, \sigma_{y_0}^2\right), \quad v_0^{(i)} \sim N\left(\bar{v}_0, \sigma_{v_0}^2\right), \quad \phi_0^{(i)} \sim N\left(\bar{\phi}_0, \sigma_{\phi_0}^2\right)$$

$$w_{x_k}^{(i)} \sim N\left(0, \sigma_{w_x}^2\right), \quad w_{y_k}^{(i)} \sim N\left(0, \sigma_{w_y}^2\right), \quad w_{v_k}^{(i)} \sim N\left(0, \sigma_{w_v}^2\right), \quad w_{\phi_k}^{(i)} \sim N\left(0, \sigma_{w_\phi}^2\right)$$

where

$$\bar{x}_0 = 120km, \quad \sigma_{x_0} = 5km, \quad \bar{y}_0 = 150km, \ \sigma_{y_0} = 5km;$$

$$\bar{v}_0 = 300\,{m}/{s}, \quad \sigma_{v_0} = 5\,{m}/{s}, \quad \bar{\phi}_0 = 30\deg, \quad \sigma_{\phi_0} = 1\deg;$$

$$\sigma_{w_x} = 5m, \quad \sigma_{w_y} = 5m, \quad \sigma_{w_v} = 1\,{m}/{s}, \quad \sigma_{w_\phi} = 0.01\deg$$

Scenario 1

Scenario 1 only has tangential accelerations.

$$at_k = \begin{cases} 0 & 0 \le k < 60 \\ 20 & 60 \le k < 90 \\ 0 \quad for & 90 \le k < 120 \ \text{m/s}^2 \\ -60 & 120 \le k < 130 \\ 0 & 130 \le k < 150 \end{cases}$$

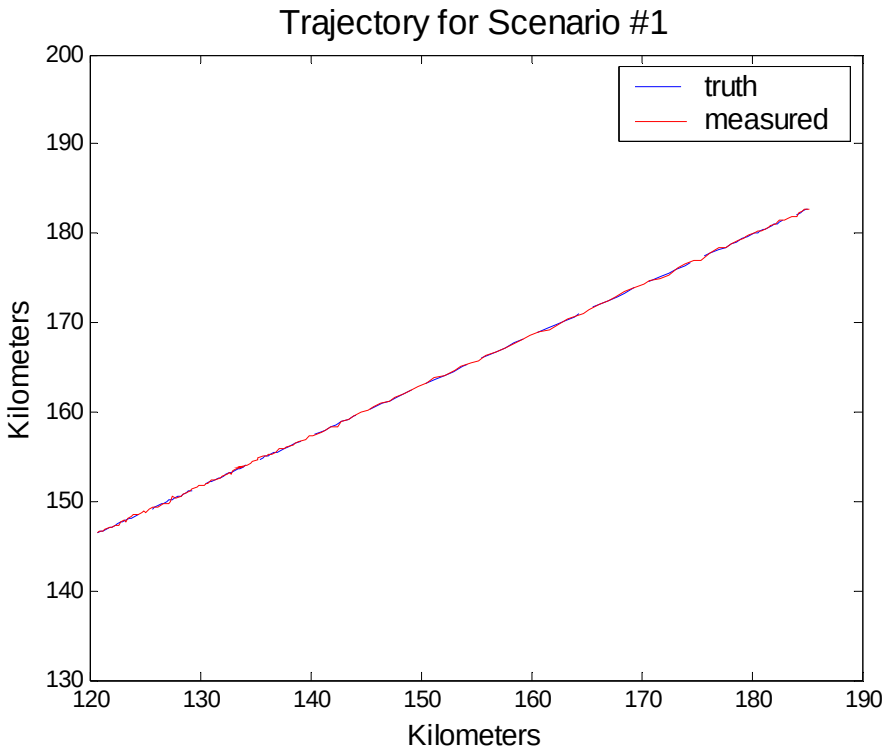

Figure 3

The figures show the set of true positions and the measurements for one run. The effect of the measurement noise is not visible due to the scale of the axes but the standard deviation of the measurement error is 140 meters from the true position.

Scenario 2

Scenario 2 only has normal accelerations.

$$an_k = \begin{cases} 0 & 0 \le k < 60 \\ 20 & 60 \le k < 90 \\ 0 & for \quad 90 \le k < 120 \\ -60 & 120 \le k < 130 \\ 0 & 130 \le k < 150 \end{cases} \text{ m/s}^2$$

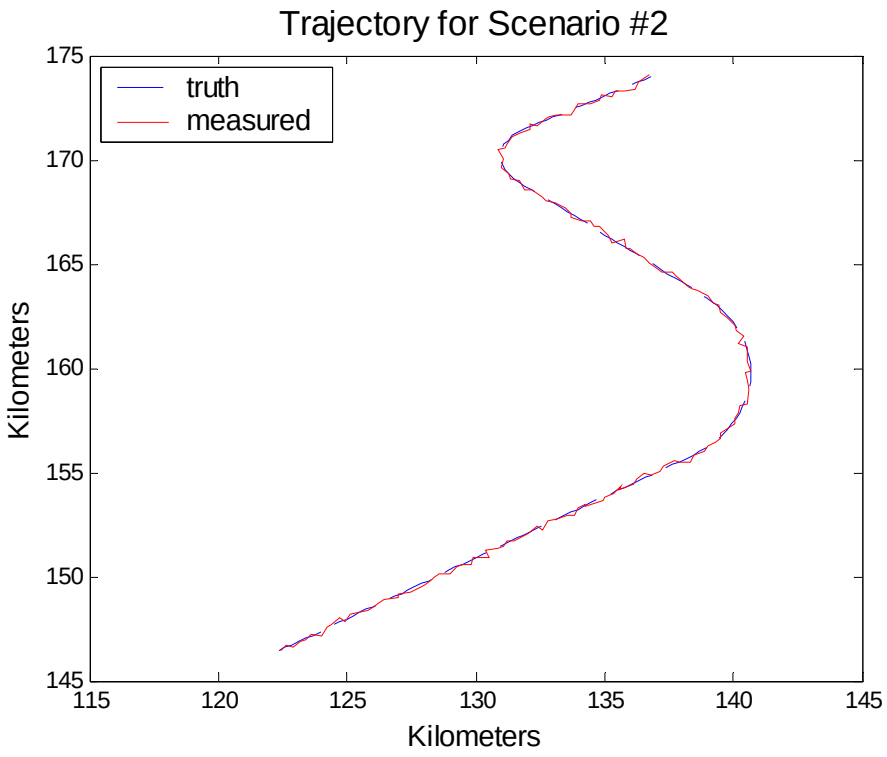

Figure 4

The accelerations in scenario 2 are symmetric to scenario 1 but are in the direction normal to the velocity.

Scenario 3

Scenario 3 consists of both normal and tangential accelerations.

$$at_k = \begin{cases} 0 & 0 \le k < 60 \\ 30 \quad for & 60 \le k < 90 \quad \text{m/s}^2 \\ 0 & 120 \le k < 130 \end{cases} \qquad an_k = \begin{cases} 0 & 0 \le k < 60 \\ 30 \quad for & 60 \le k < 90 \quad \text{m/s}^2 \\ 0 & 120 \le k < 130 \end{cases}$$

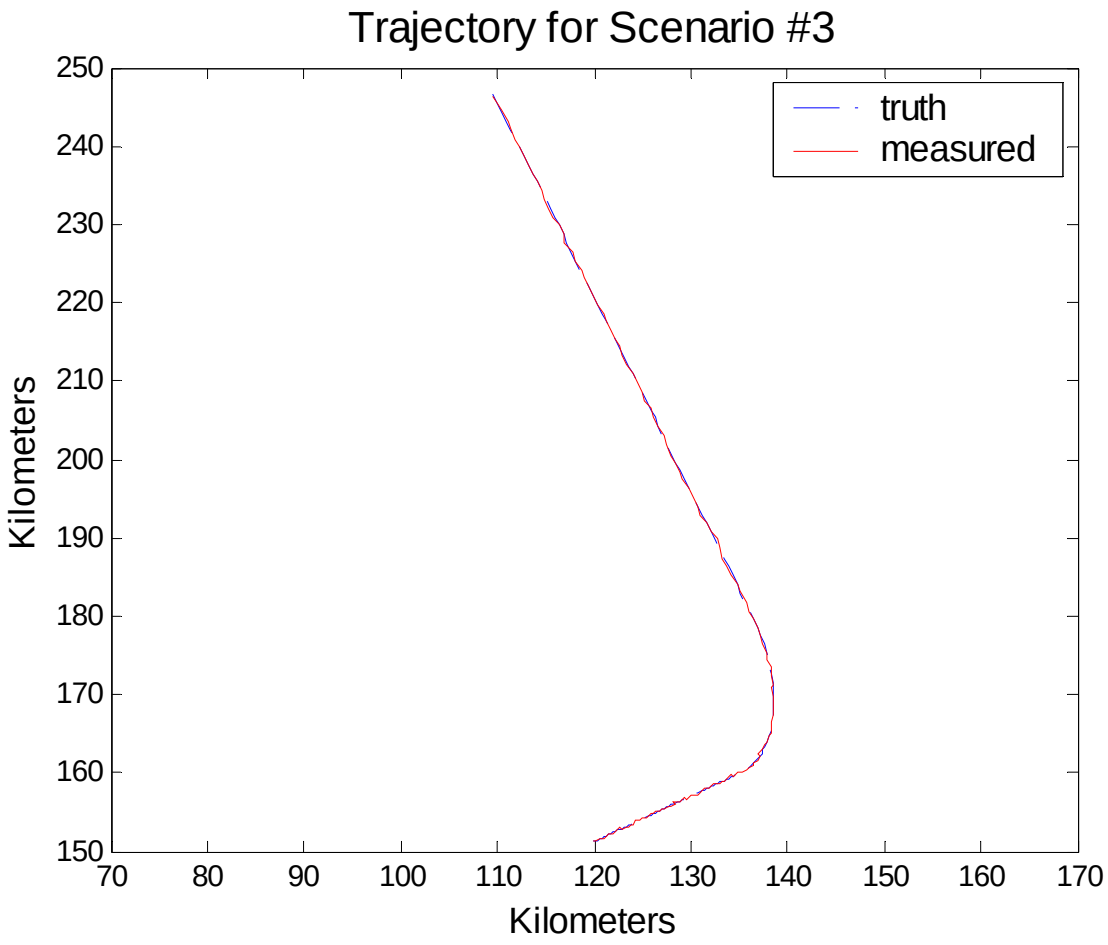

Figure 5

Scenario 3 will be more difficult to for the algorithms to track because the model set does not contain a single model that accounts for two types of accelerations in a single maneuver. Therefore, this scenario will test how well the models in each algorithm cooperate.

Position measurements were taken in Cartesian coordinates with sampling time, $T = 1s$, providing measurements z_k, k = 1, 2, . . . of the target.

$$z_k = \begin{bmatrix} z_x \\ z_y \end{bmatrix}_k = \begin{bmatrix} x + v_x \\ y + v_y \end{bmatrix}_k$$

The measurement noise, $v_k = \begin{bmatrix} v_x \\ v_y \end{bmatrix}_k$, has the following Gaussian distribution:

$$v \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 100^2 & 0 \\ 0 & 100^2 \end{bmatrix}\right) meters .$$

The algorithms to be compared will track a target in two-dimensional Cartesian coordinates instead of in polar coordinates because polar coordinates introduce nonlinearities to the measurement that are beyond the scope of this comparison. This decision was made ignoring the fact that most tracking is done using radar in polar coordinates

5. METHODS OF COMPARISON

The MM algorithms were compared. The state estimates were then compared to the ground truths and the root mean square error (RMSE) was taken for both the position and for the velocity. RMSE is calculated as follows

$$Position\ RMSE_k = \left(\frac{1}{N} \sum_{i=1}^{N} \left[\left(x_k^{(i)} - \hat{x}_k^{(i)} \right)^2 + \left(y_k^{(i)} - \hat{y}_k^{(i)} \right)^2 \right] \right)^{1/2}$$

$$Velocity\ RMSE_k = \left(\frac{1}{N} \sum_{i=1}^{N} \left[\left(\dot{x}_k^{(i)} - \hat{\dot{x}}_k^{(i)} \right)^2 + \left(\dot{y}_k^{(i)} - \hat{\dot{y}}_k^{(i)} \right)^2 \right] \right)^{1/2}$$

where N is the number of runs, $\left(x_k^{(i)}, y_k^{(i)}, \dot{x}_k^{(i)}, \dot{y}_k^{(i)} \right)$ and $\left(\hat{x}_k^{(i)}, \hat{y}_k^{(i)}, \hat{\dot{x}}_k^{(i)}, \hat{\dot{y}}_k^{(i)} \right)$ denote the true and the estimated states, respectively, in run *i* at time *k*.

Each of the algorithm's credibility is measured in terms of log average normalized estimation error squared (LNEES). LNEES is calculated as follows

$$LNEES_k = 10 \log \left[\frac{1}{4N} \sum_{i=1}^{N} \left(x_k^{(i)} - \hat{x}_k^{(i)} \right)^T P_k^{-1} \left(x_k^{(i)} - \hat{x}_k^{(i)} \right) \right]$$

where $x = \left(x_k^{(i)}, y_k^{(i)}, \dot{x}_k^{(i)}, \dot{y}_k^{(i)} \right)^T$ and $\hat{x} = \left(\hat{x}_k^{(i)}, \hat{y}_k^{(i)}, \hat{\dot{x}}_k^{(i)}, \hat{\dot{y}}_k^{(i)} \right)^T$.

Computational complexity is a measure of the amount of time needed for an algorithm to run. The longer an algorithm takes to run the more computationally complex it is. The computational complexity was calculated using MATLAB's stopwatch timer. The complexities were then normalized relative to AMM and AMM would have a computational complexity of one.

6. RESULTS

RMSE has no physical interpretation but it does have a probabilistic meaning in that it is similar to the standard deviation of the error [1]. It also implicitly favors conditional mean based estimator such as the Kalman filter since conditional mean minimizes standard error [1]. Log average normalized estimation error squared (LNEES) is a measure of the credibility of a filter [4]. A perfectly credible filter will have an LNEES value of zero. While a filter with a negative LNEES value is considered pessimistic because the error is better than the filter thinks that it is. A filter with a positive LNEES is considered optimistic because the filter thinks that the estimate is better than it actually is.

The presentation of the results will begin with the computational complexity of the algorithms followed by plots that are arranged according to test scenario. The first three graphs for each scenario show position RMSE, velocity RMSE, and LNEES plots, respectively, with all of the tracking algorithms plotted together in each of the figures. The subsequent plots show the mode probabilities for each MM target-tracking algorithm. All of the plots show the average over the 100 runs. The reader is referred to the **Test Scenarios** section for a description of the scenarios.

Normalized Computational Complexities

Algorithm	**AMM**	**GPB1**	**GPB2**	**IMM**	**RIMM**	**B-BEST**	**VA**
Complexity	1.00	1.09	7.91	2.3	4.64	6.67	7.23

Table 8

Scenario 1:

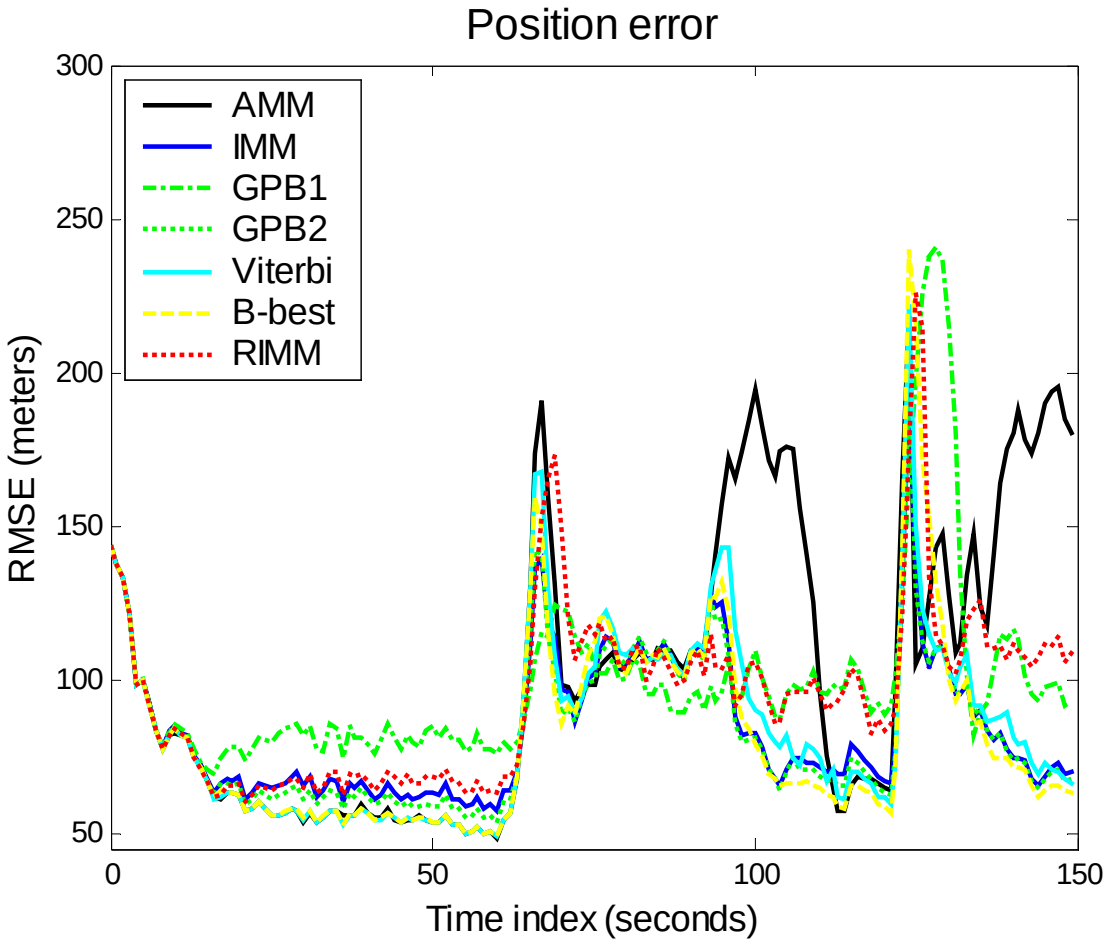

Figure 6: Position RMSE for the 7 algorithms

Figure 6 shows that B-Best, AMM, and the Viterbi algorithm have the best steady-state RMSE followed by GPB2, IMM, RIMM, and finally GPB1. The worst peak errors are as follows: AMM, RIMM, Viterbi, B-Best, GPB1, GPB2, and then IMM.

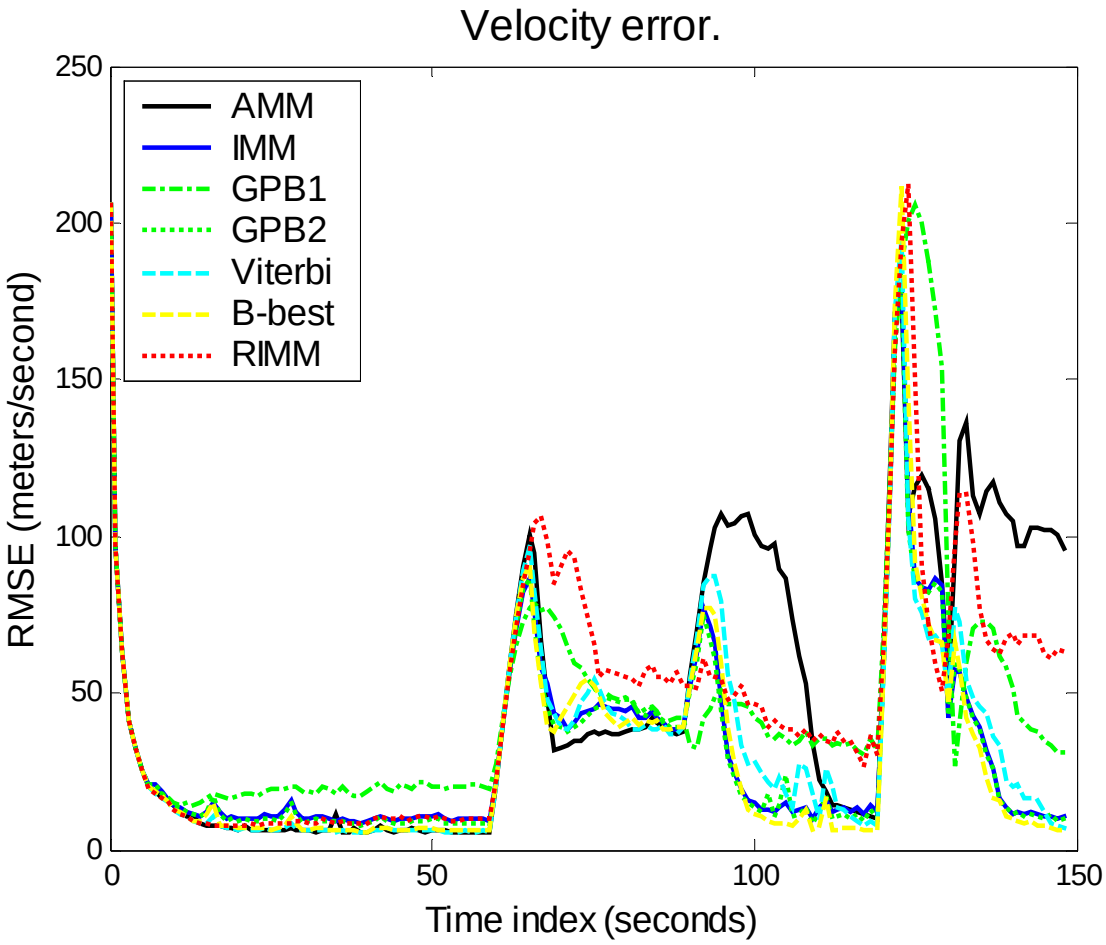

Figure 7: The RMSE of the velocity estimates

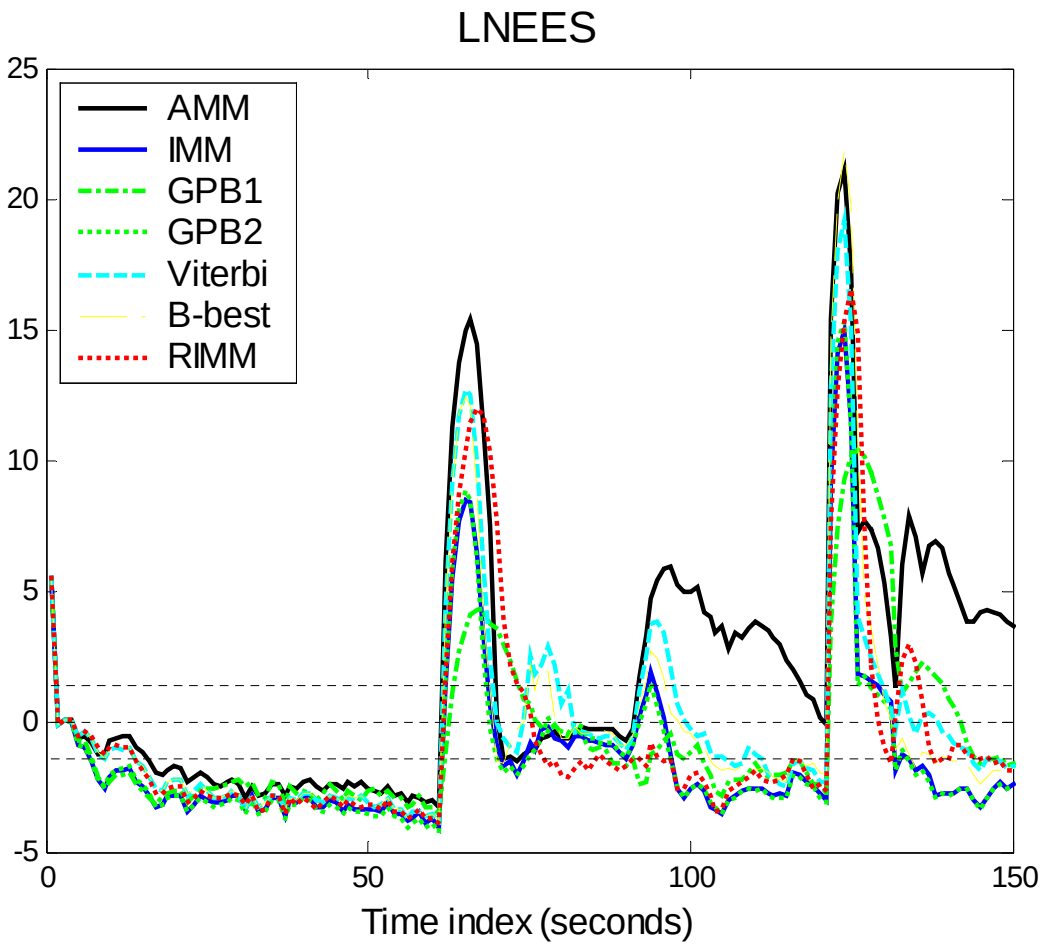

Figure 8: The LNEES credibility measure for the 7 algorithms

The horizontal line at zero represents perfect credibility and horizontal lines at $y = -1.4363$ and $y = 1.3389$ are the boundaries of the 95% confidence interval.

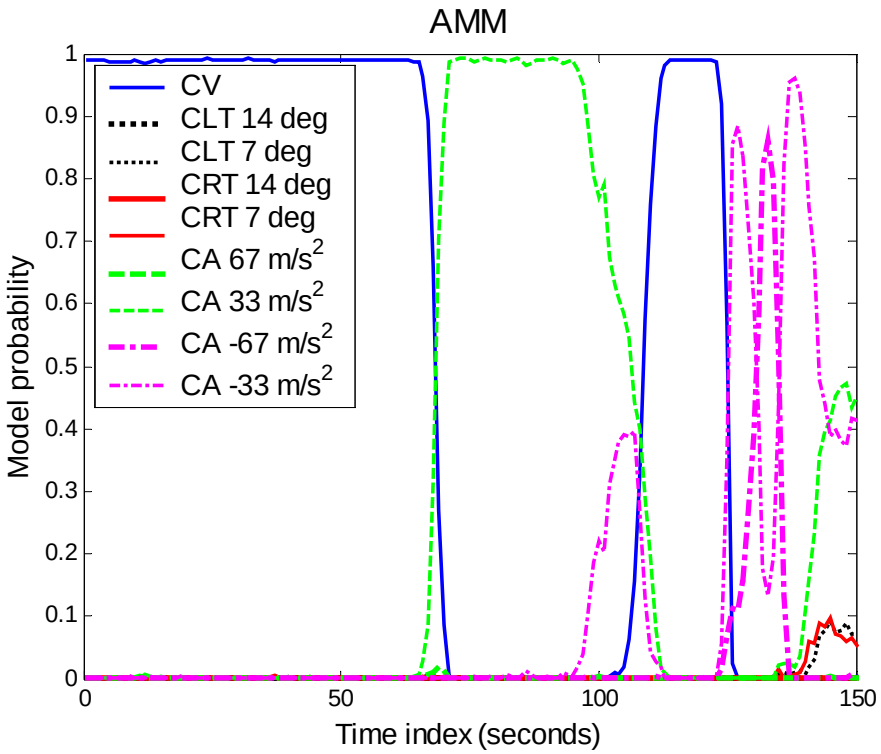

Figure 9: The 9 model probabilities of AMM

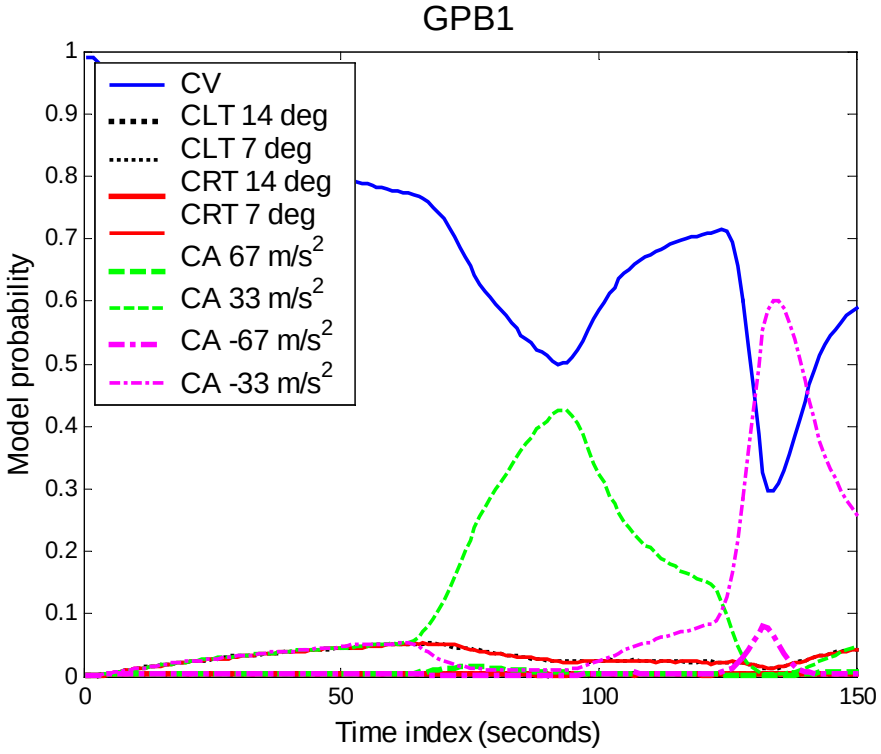

Figure 10: The 9 model probabilities of GPB1

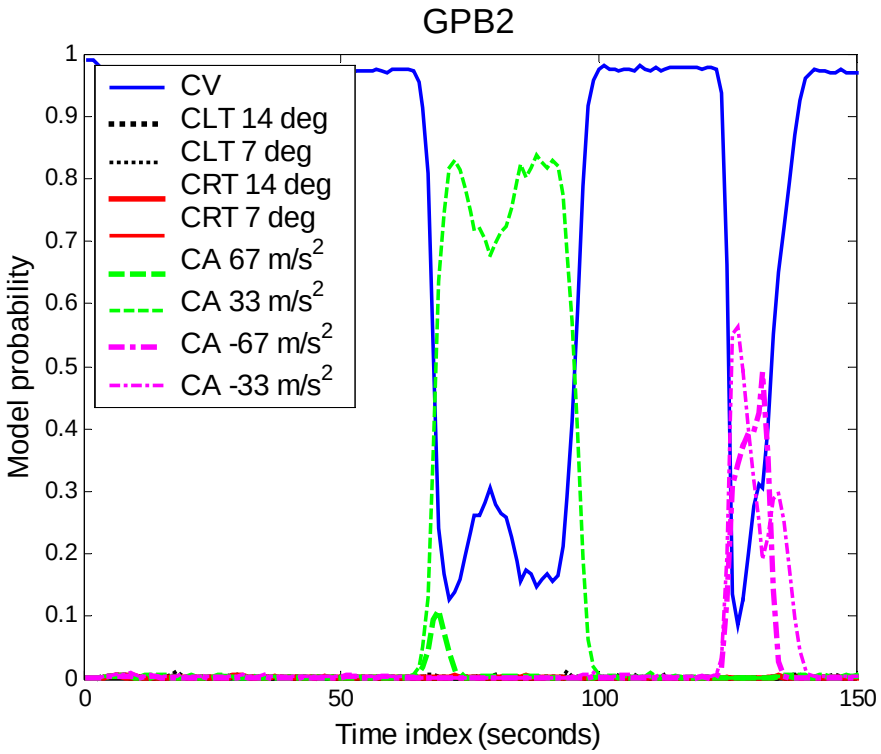

Figure 11: The 9 model probabilities of GPB2

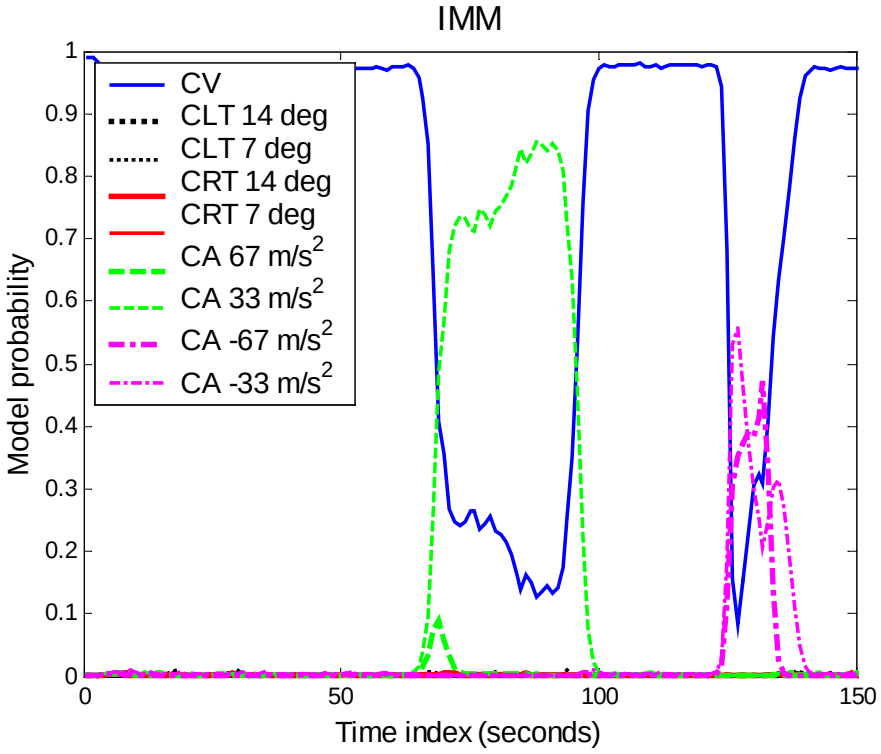

Figure 12: The 9 model probabilities of IMM

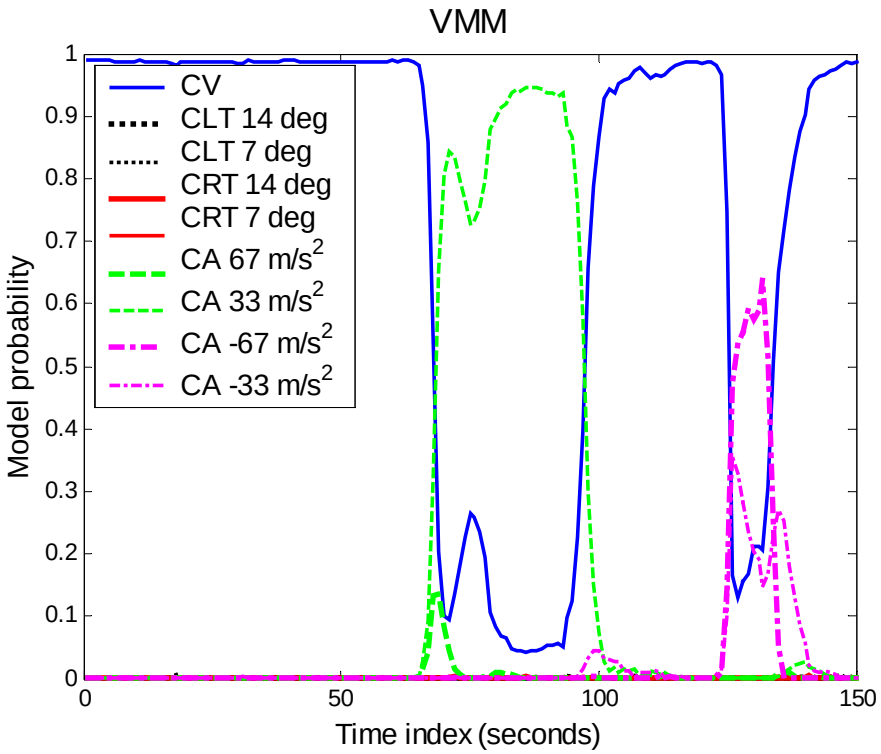

Figure 13: The 9 model probabilities of the Viterbi Algorithm

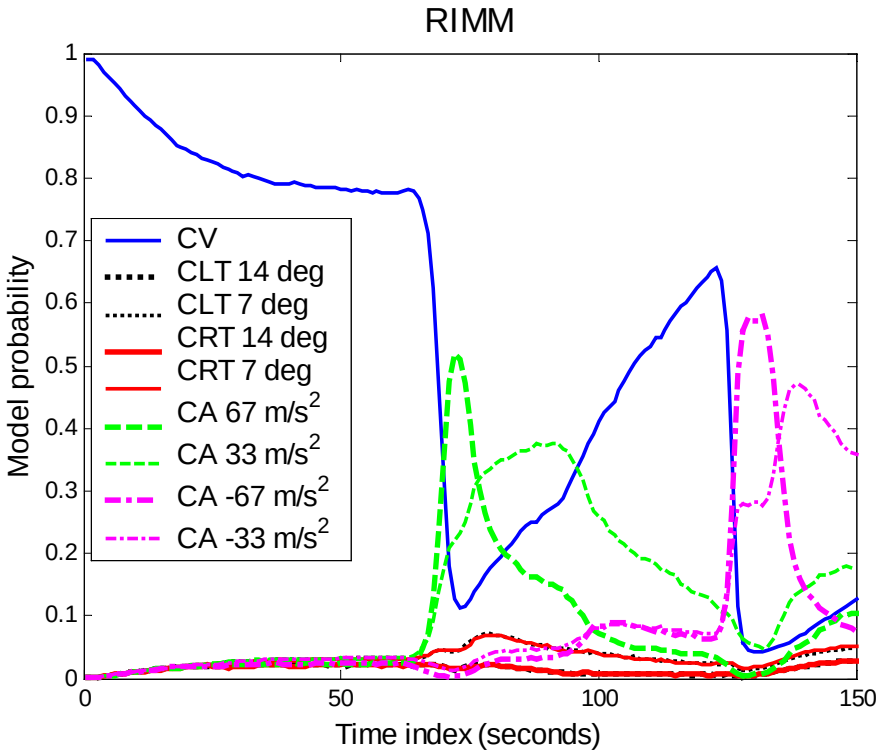

Figure 14: The 9 model probabilities of RIMM

Scenario 2:

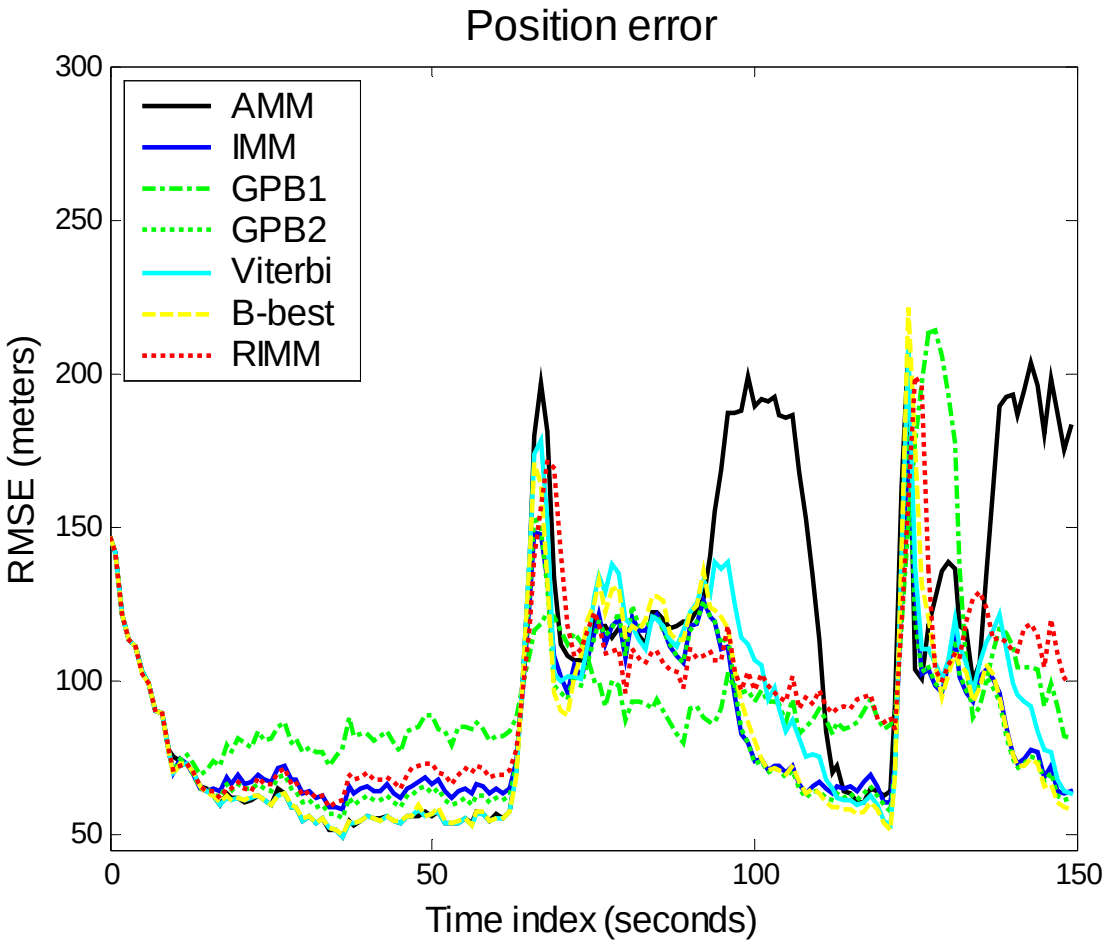

Figure 15

This figure demonstrates that B-Best, AMM, and the Viterbi algorithm have the best steady-state RMSE followed by GPB2, IMM, RIMM, and finally GPB1. The worst peak errors are as follows: AMM, Viterbi, B-Best, RIMM, GPB1, GPB2, and then IMM.

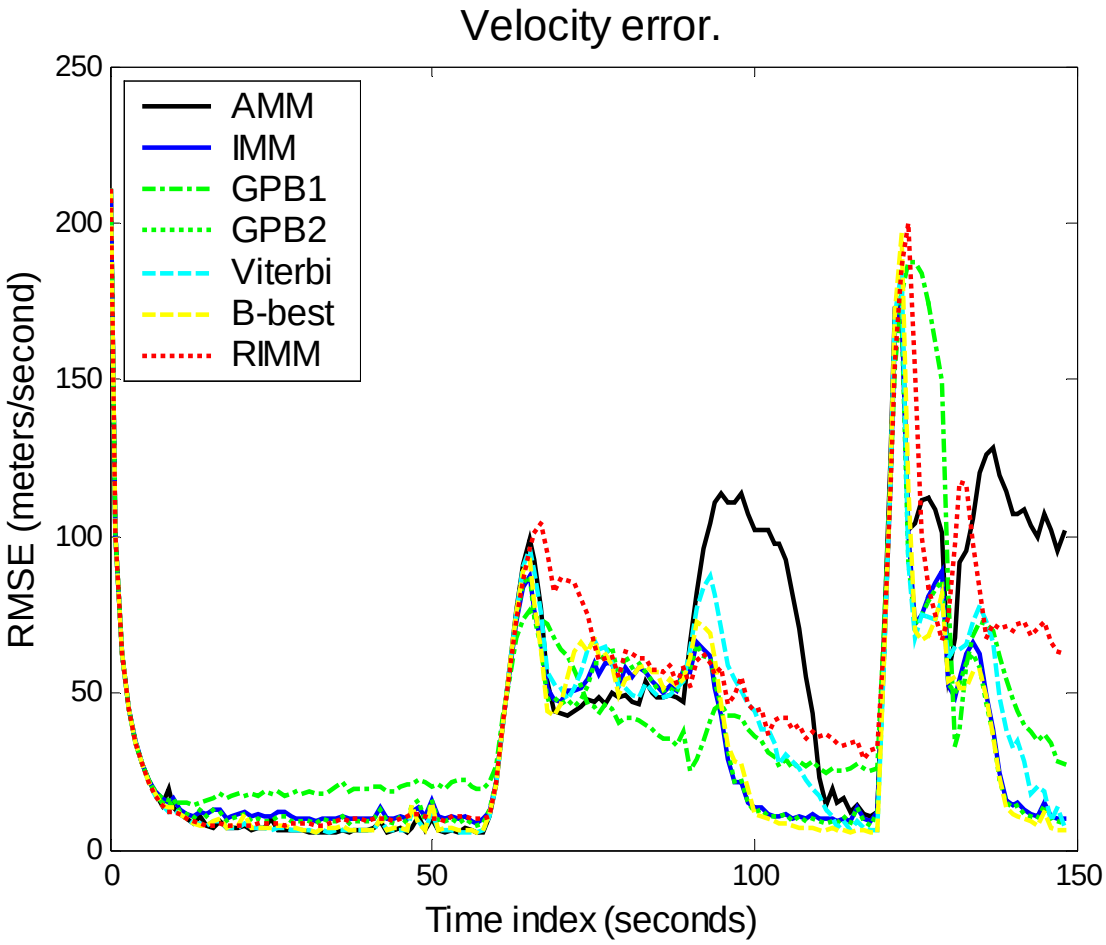

Figure 16: Velocity RMSE of the 7 algorithms

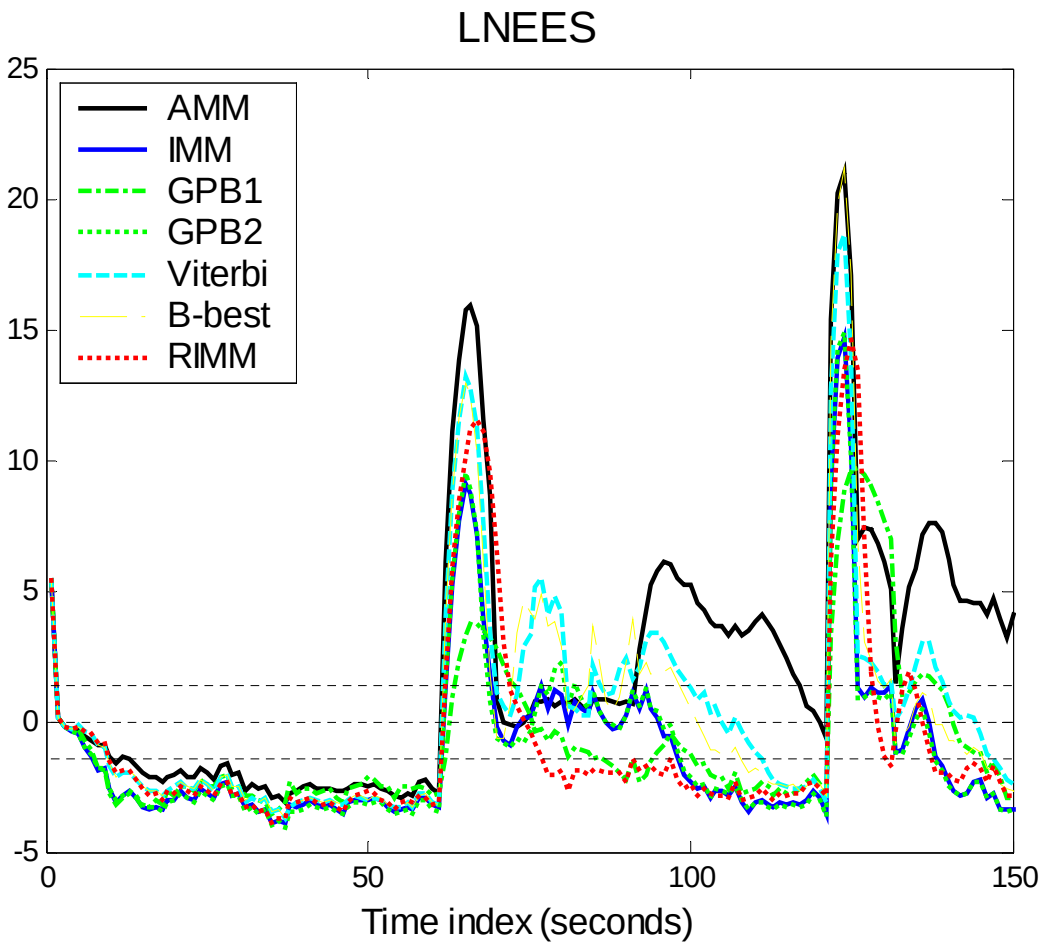

Figure 17: The LNEES credibility measure for the 7 algorithms

The horizontal line at zero represents perfect credibility and horizontal lines at $y = -1.4363$ and $y = 1.3389$ are the boundaries of the 95% confidence interval.

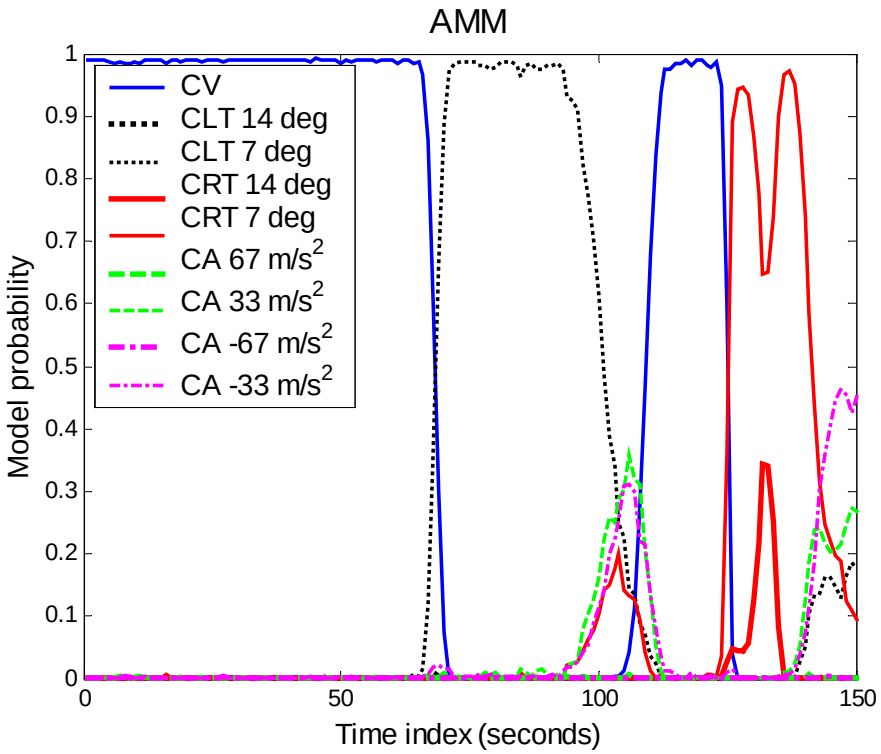

Figure 18: The 9 model probabilities of AMM

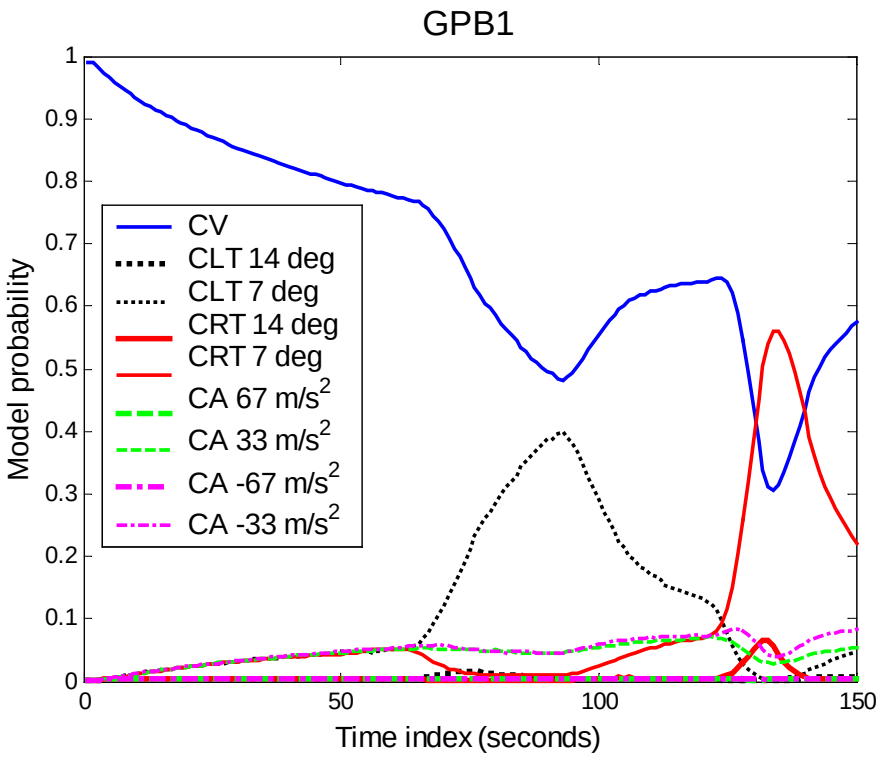

Figure 19: The 9 model probabilities of GPB1

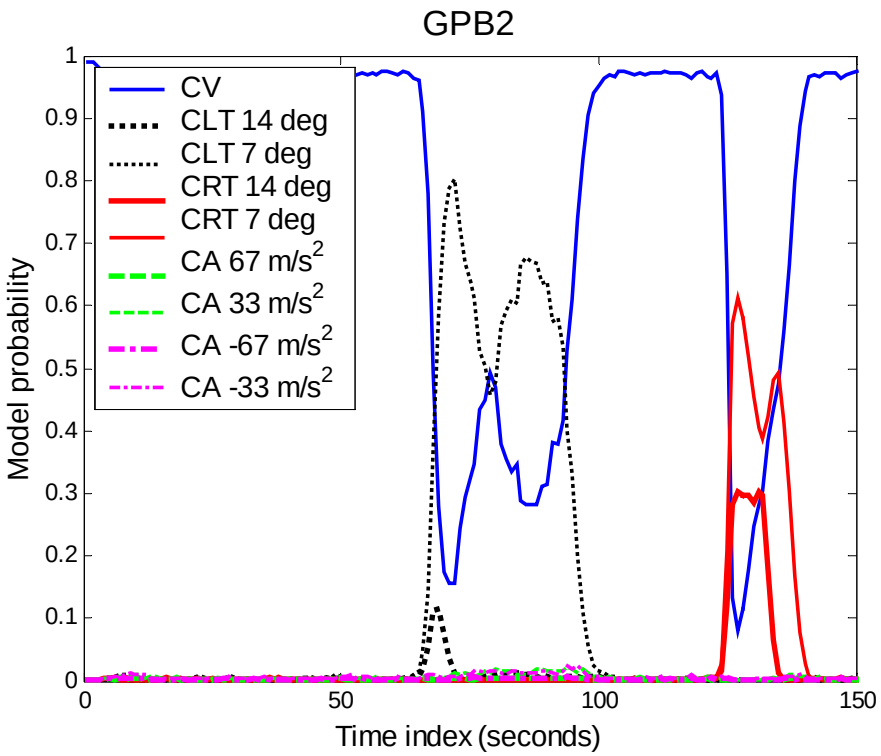

Figure 20: The 9 model probabilities of GPB2

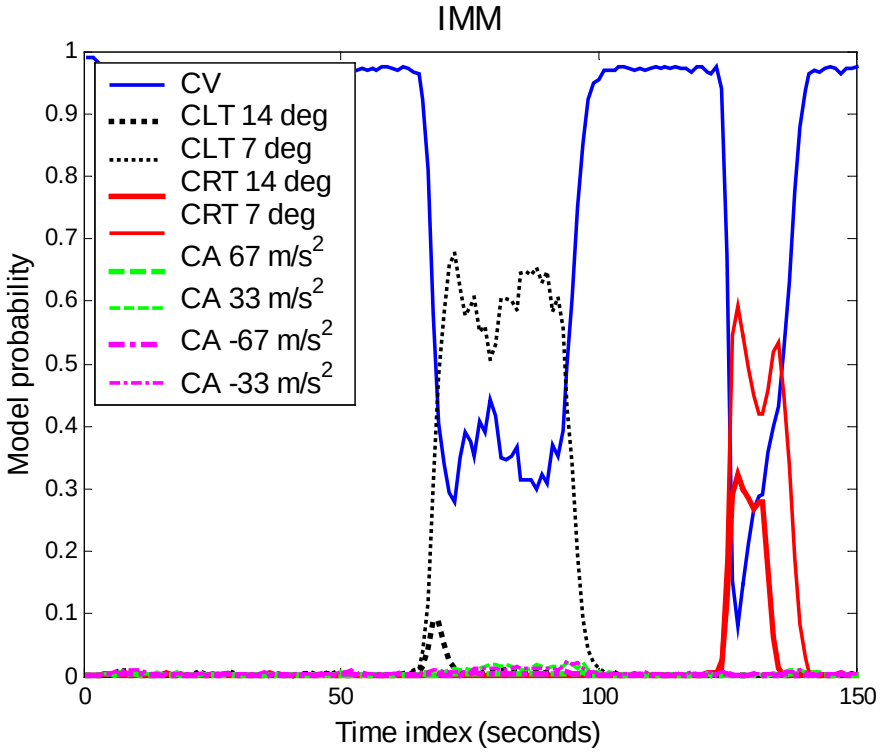

Figure 21: The 9 model probabilities of IMM

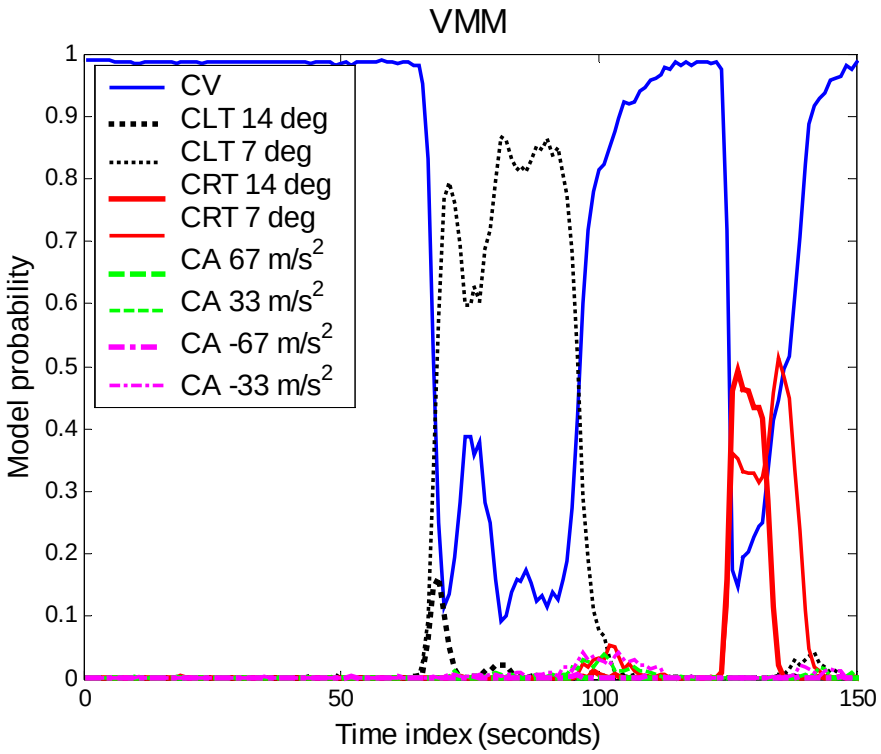

Figure 22: The 9 model probabilities of the Viterbi Algorithm

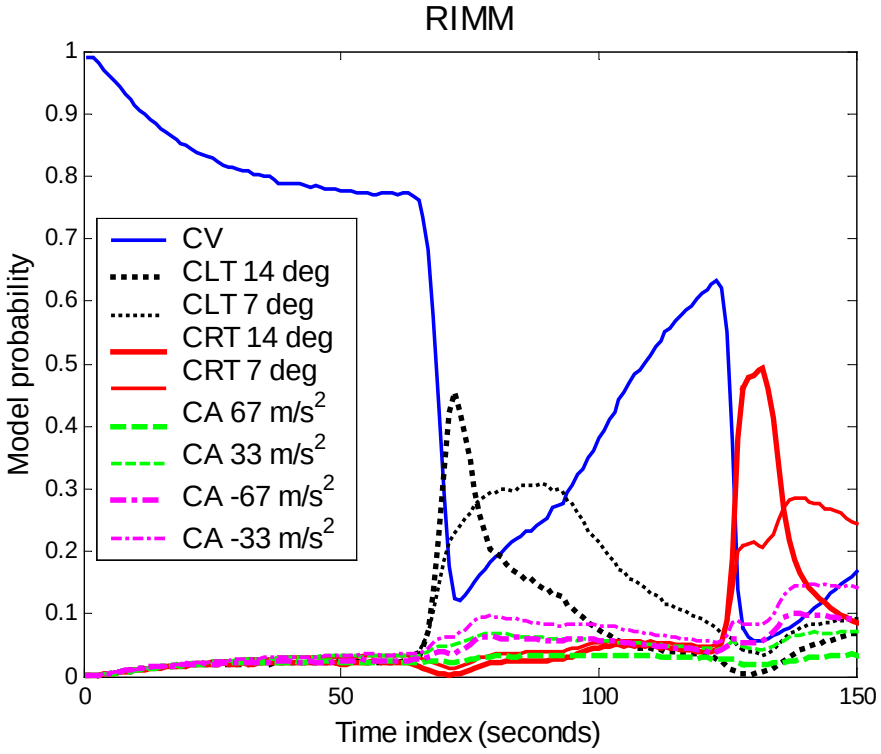

Figure 23: The 9 model probabilities of RIMM

Scenario 3:

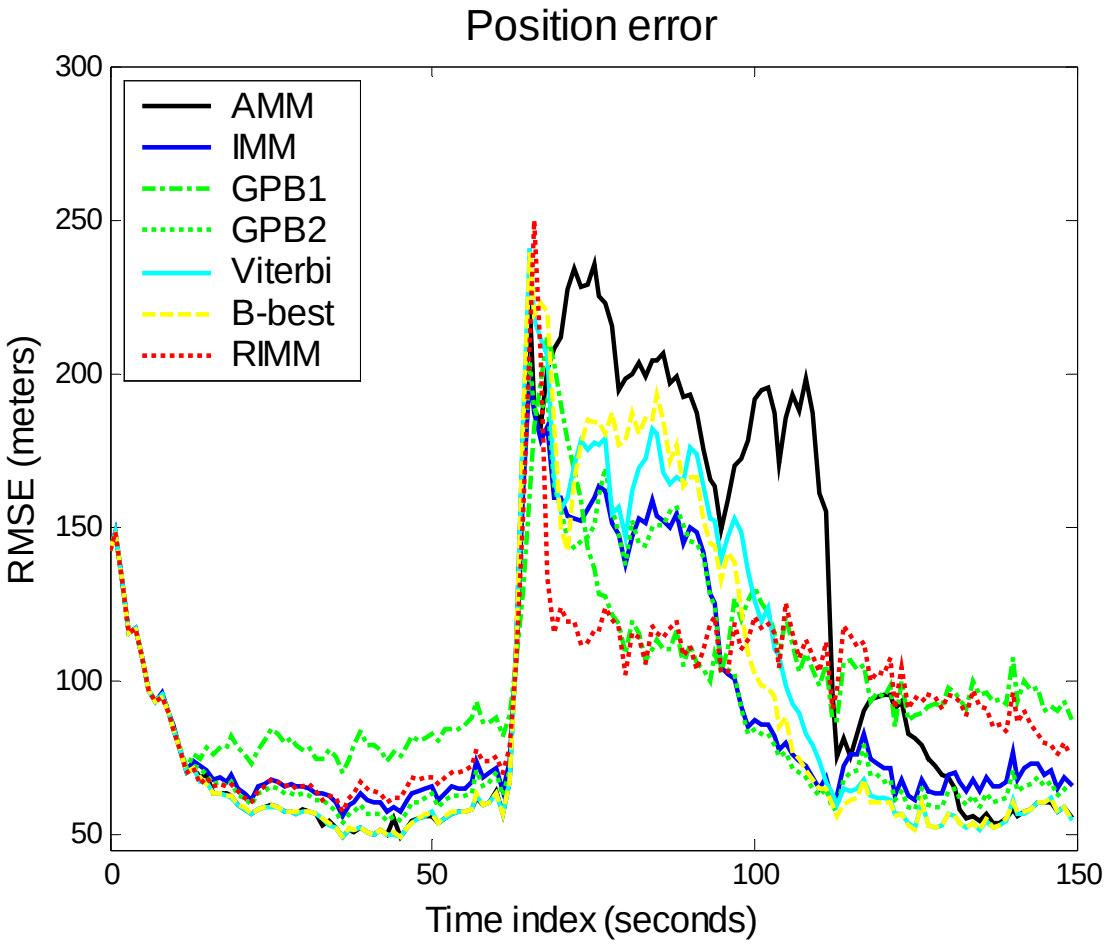

Figure 24: Position RMSE of the 7 algorithms

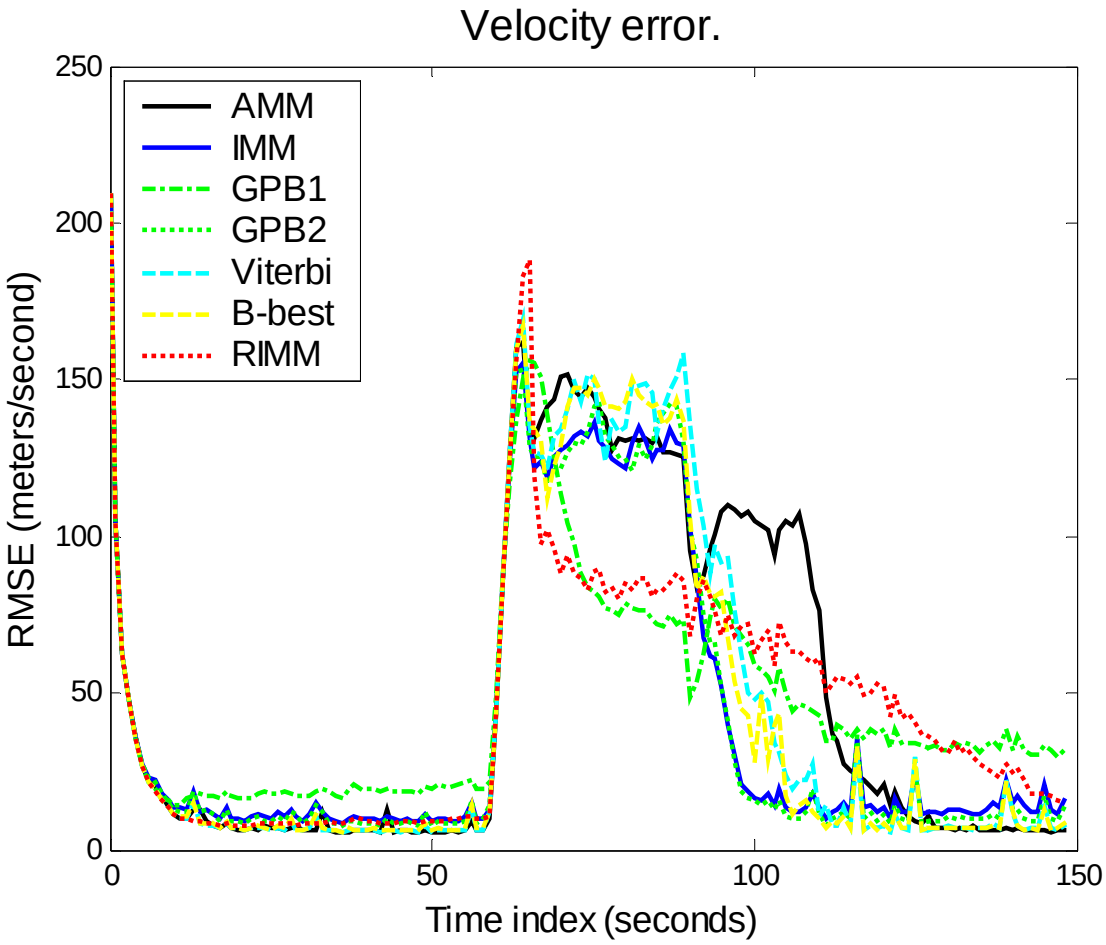

Figure 25: Velocity RMSE of the 7 algorithms

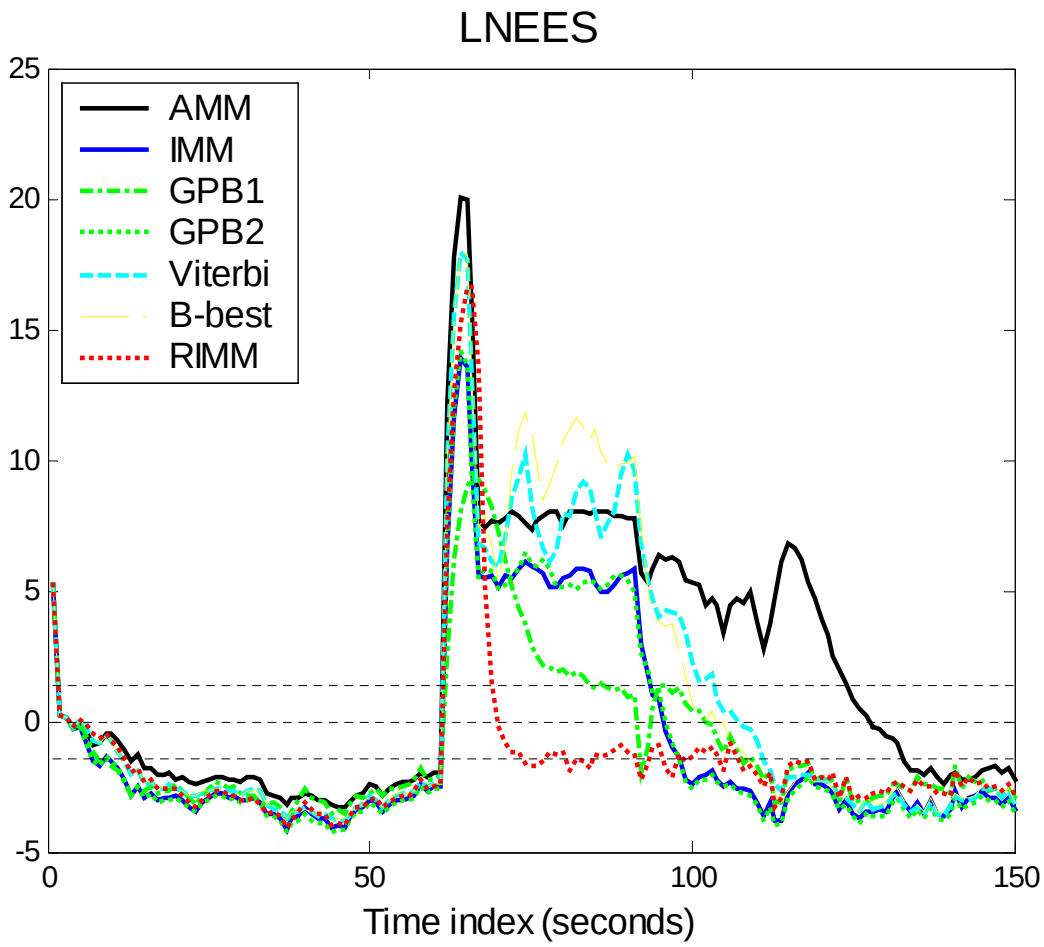

Figure 26: The LNEES credibility measure for the 7 algorithms

The horizontal line at zero represents perfect credibility and horizontal lines at $y = -1.4363$ and $y = 1.3389$ are the boundaries of the 95% confidence interval.

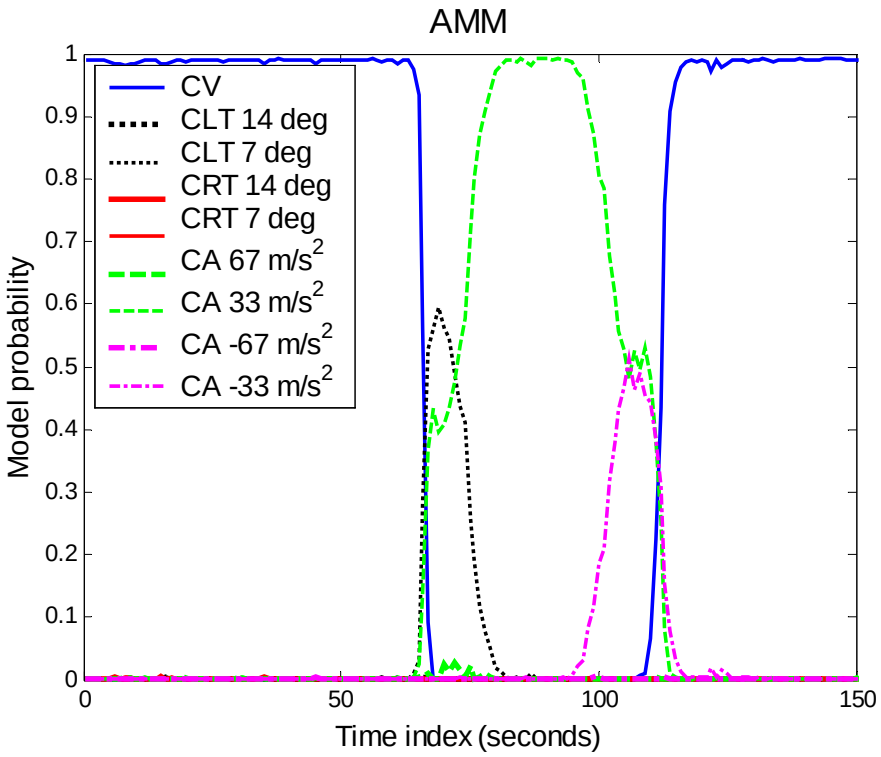

Figure 27: The 9 model probabilities of AMM

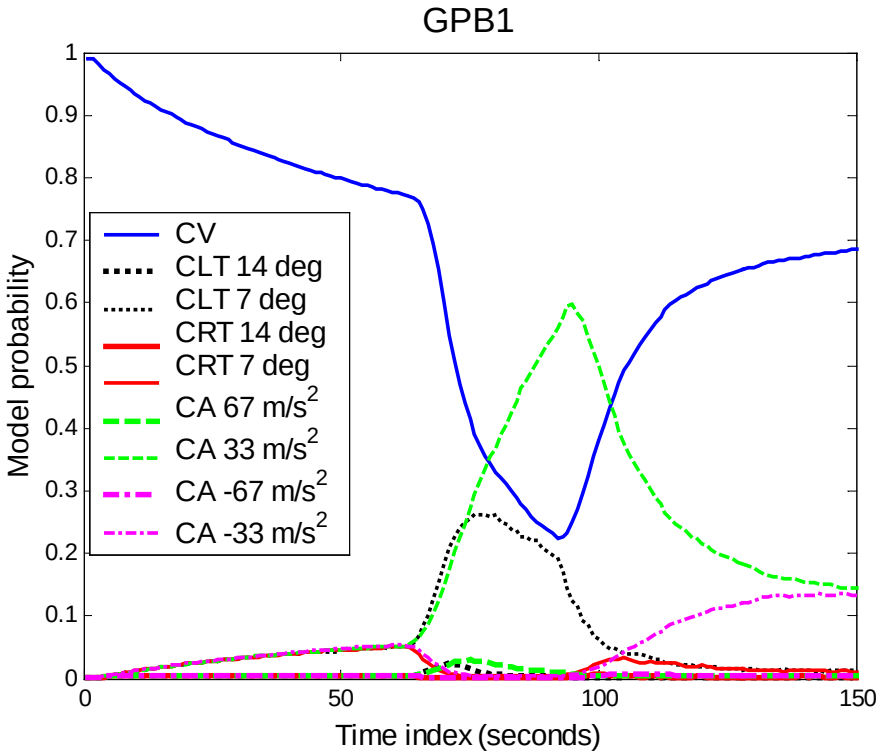

Figure 28: The 9 model probabilities of GPB1

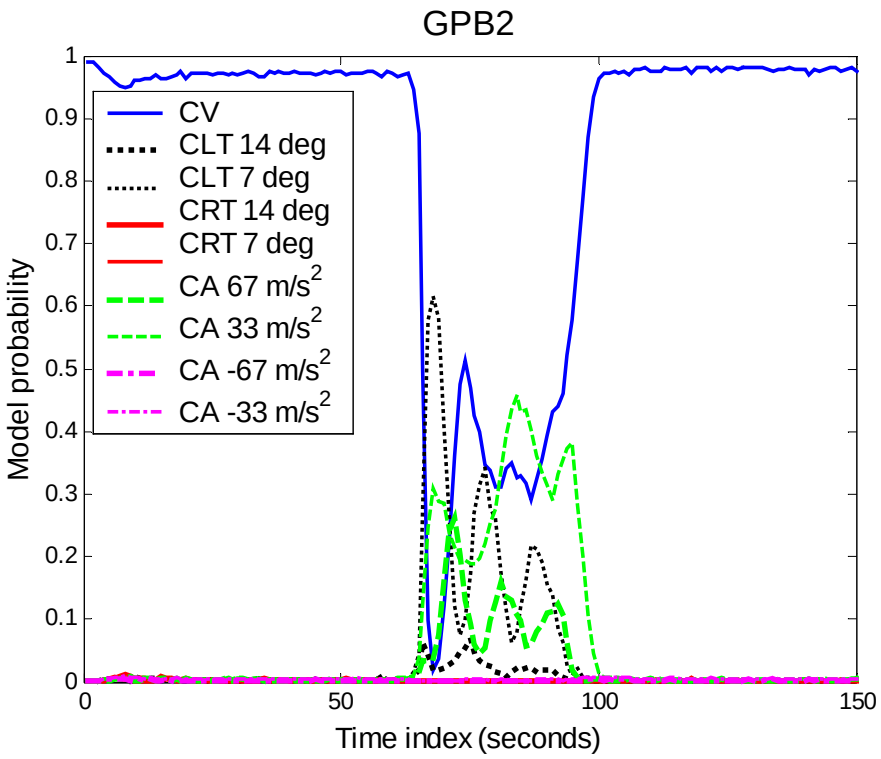

Figure 29: The 9 model probabilities of GPB2

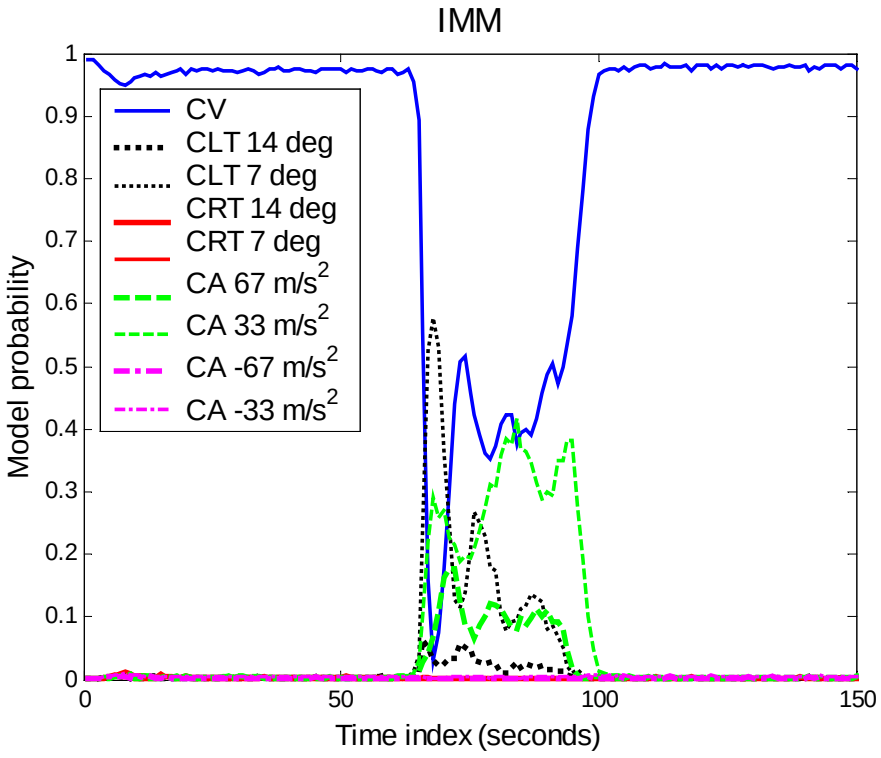

Figure 30: The 9 model probabilities of IMM

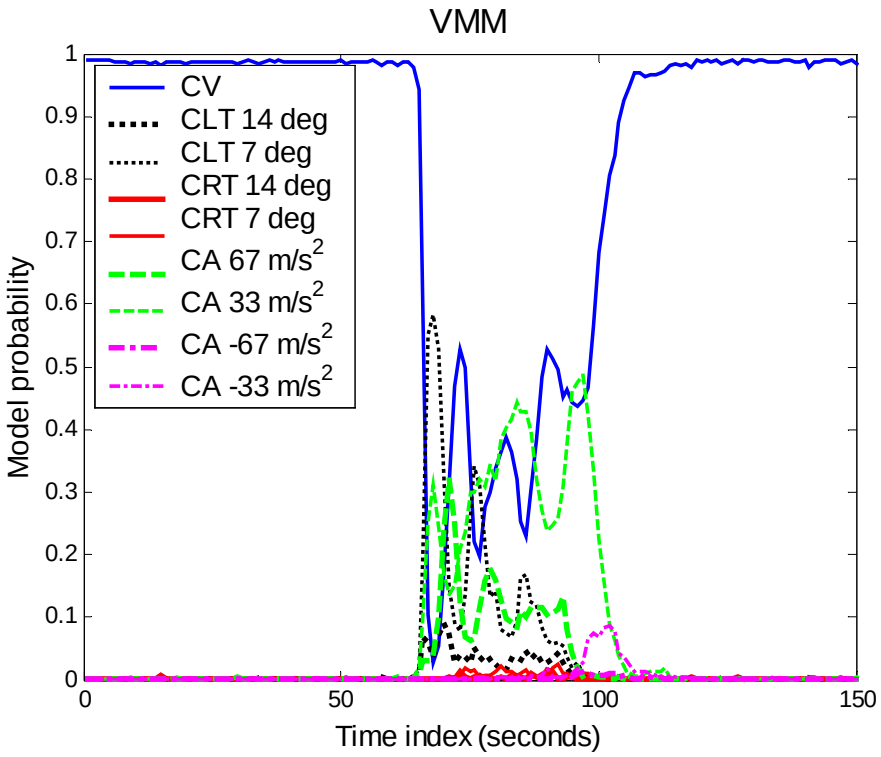

Figure 31: The 9 model probabilities of the Viterbi Algorithm

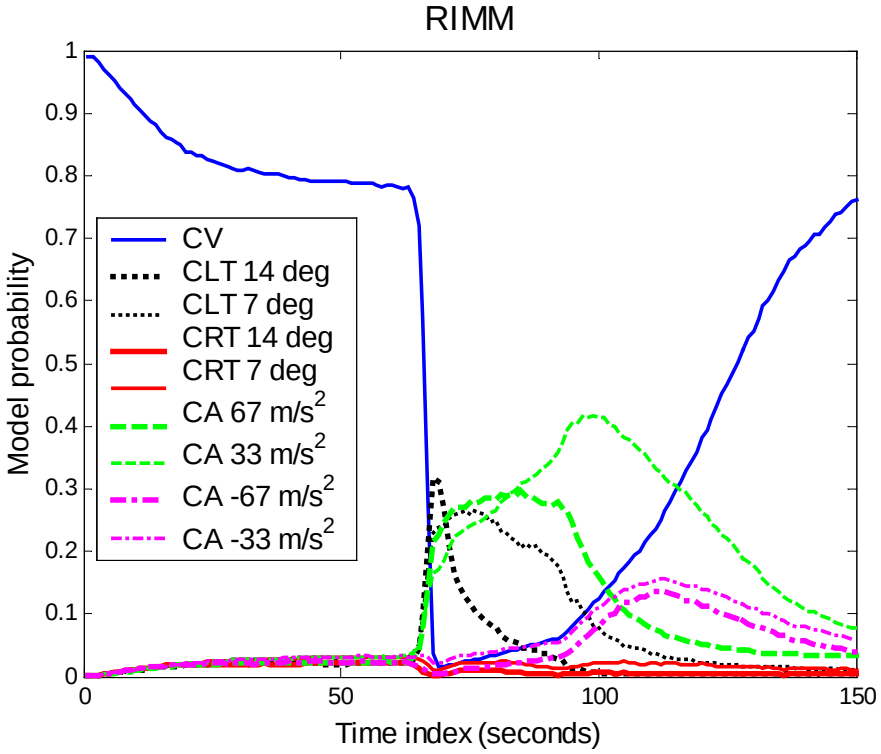

Figure 32: The 9 model probabilities of RIMM

7. DISCUSSIONS

GPB1 is neither sensitive to the selection of *Q* nor to transition in mode. This insensitivity is a desirable characteristic but unfortunately the RMSE for GPB1 was not as small as the rest of the algorithms when CV was the true mode. GPB1 did however have a competitive RMSE during a maneuver. GPB1's slow response was originally unsettling because it does not make practical sense. GPB1 is an established algorithm and it was hard to believe that GPB1 would not converge to a common RMSE with the rest of the algorithms. Then this question had to be answered, why does it take so long for the CV model probability to dominate the other model probabilities in the model probability diagram? See Figures 10, 19, 28, and 51.

The model likelihood values for GPB1 were discovered not to vary much from each other. Their likelihoods were so close to each other because of the way that GPB1 reinitializes its filters, which is to let each filter start with the same value at the beginning of each iteration. This reinitialization does not allow the CV model to dominate as much in the other algorithms during a non-maneuver.

Once all of the filters have been updated using this initialization, all of the position estimates lie within what I call the measurement noise envelope. This envelope has a radius of about 140 meters, which is the standard deviation of the measurement error. Therefore, when the true model is the constant velocity model, the filter can easily mistake the measurement noise for a maneuver. When the target is maneuvering, GPB1 does not suffer as badly from the noise envelope because the model probability that represents the maneuver is able to accumulate faster than the

other model probabilities. This is because the other model updates will have lower likelihoods due to the greater model-maneuver mismatch. To confirm this, GPB1 was run using scenario 1 and a measurement noise with smaller variance, $R = \begin{bmatrix} 25^2 & 0 \\ 0 & 25^2 \end{bmatrix}$. There was not much improvement in terms of RMSE but there was a twofold improvement in the response of all of the model probabilities.

Future works include an improvement to my model set by using a model that can use its velocity estimate to adapt the turn rate used in the constant turn models. I think that I should also investigate using the overall velocity estimate to estimate the direction of the velocity instead of using a model's velocity estimate for my constant acceleration model.

8. CONCLUSIONS

IMM could be interpreted as the best target-tracking algorithm examined here but there is no clear-cut best algorithm based on the results from this comparison. IMM and the rest of the algorithms had comparable RMSE values for all of the scenarios but IMM, however, achieved it much more efficiently. Additionally, IMM's RMSE values converged as fast as the rest of the algorithms presented in this paper.

The characteristic of interest when selecting an algorithm is the ratio of efficiency to RMSE. Naturally, the smallest possible RMSE is desirable but it is not always feasible to meet the necessary computational requirements. If the computational resources are available to use B-Best, then B-best is the best choice. Otherwise, IMM should be used because its RMSE is competitively low and it requires much less computation. GPB2 performed only slightly better than IMM in the RMSE sense but its computational complexity was the most demanding compared to all of the algorithms presented here. GPB1 is not a good choice because the average RMSE is much higher than the other algorithms when the target is not maneuvering. AMM performs with the least amount of computation but has the longest recovery time after a maneuver. The Viterbi algorithm was expensive computationally and had a slower transition between modes compared to B-Best but it has one of the lower RMSEs. RIMM did not perform as well as the other filters because of its slow transition after a change in mode and its computational complexity was the median value of the complexities presented here.

REFERENCES

[1] X. Rong Li, *Applied Estimation and Filtering, course notes.* University of New Orleans. 2003

[2] X. Rong Li and Vesselin P. Jilkov, "A Survey of Maneuvering Target Tracking – Part V: Multiple-Model Methods". Submitted for publication 2003, URL: http://ece.engr.uno.edu/isl/MTTSurveys.htm

[3] X. Rong Li and Vesselin P. Jilkov, "A Survey of Maneuvering Target Tracking – Part I: Dynamic Models". Released for publication 2003, URL: http://ece.engr.uno.edu/isl/MTTSurveys.htm

[4] X. Rong Li, Zhanlue Zhao, and Vesselin P. Jilkov, "Estimator's credibility and its measures". Proceedings of International Federation of Automatic Control 15th World Congress, Barcelona, Spain, July 2002 URL: http://ece.engr.uno.edu/isl/MTTSurveys.htm

APPENDIX

One cycle of the AMM algorithm is as follows:

1. Model-conditioned filtering (for i =1,2,. . .,M):

Predicted state: $\hat{x}_{k|k-1}^{(i)} = F_{k-1}^{(i)} \hat{x}_{k-1|k-1}^{(i)} + G_{k-1}^{(i)} u_{k-1}^{(i)} + \Gamma_{k-1}^{(i)} \overline{w}_{k-1}^{(i)}$

Predicted measurement: $\hat{z}_{k|k-1}^{(i)} = H_k^{(i)} \hat{x}_{k|k-1}^{(i)} + \overline{v}_k^{(i)}$

Measurement residual: $\tilde{z}_k^{(i)} = z_k - \hat{z}_{k|k-1}^{(i)}$

Predicted covariance: $P_{k|k-1}^{(i)} = F_{k-1}^{(i)} P_{k-1|k-1}^{(i)} \left(F_{k-1|k-1}^{(i)}\right)^T + G_{k-1}^{(i)} Q_{k-1}^{(i)} \left(G_{k-1}^{(i)}\right)^T$

Residual covariance: $S_k^{(i)} = H_k^{(i)} P_{k|k-1}^{(i)} \left(H_k^{(i)}\right)^T + R_k^{(i)}$

Filter gain: $K_k^{(i)} = P_{k|k-1}^{(i)} \left(H_k^{(i)}\right)^T \left(S_k^{(i)}\right)^{-1}$

Updated state: $\hat{x}_{k|k}^{(i)} = \hat{x}_{k|k-1}^{(i)} + K_k^{(i)} \tilde{z}_k^{(i)}$

Updated covariance: $P_{k|k}^{(i)} = P_{k|k-1}^{(i)} - K_k^{(i)} S_k^{(i)} \left(K_k^{(i)}\right)^T$

2. Mode probability update (for i =1,2,. . .,M):

Model likelihood: $L_k^{(i)} \triangleq p\left[\tilde{z}_k^{(i)} \mid m_{(i)}^k, z^{k-1}\right] = \dfrac{\exp\left[-\frac{1}{2}\left(\tilde{z}_k^{(i)}\right)' \left(S_k^{(i)}\right)^{-1} \tilde{z}_k^{(i)}\right]}{\left|2\pi S_k^{(i)}\right|^{1/2}}$

Mode probability: $\mu_k^{(i)} = \dfrac{\mu_{k-1}^{(i)} L_k^{(i)}}{\sum_{j \in M} \mu_{k-1}^{(j)} L_k^{(j)}}$

3. Estimate fusion:

Overall estimate: $\hat{x}_{k|k} = \sum_{i \in M} \hat{x}_{k|k}^{(i)} \mu_k^{(i)}$

Overall covariance: $P_{k|k} = \sum_{i \in M} \left[P_{k|k}^{(i)} + \left(\hat{x}_{k|k} - \hat{x}_{k|k}^{(i)}\right)\left(\hat{x}_{k|k} - \hat{x}_{k|k}^{(i)}\right)'\right] \mu_k^{(i)}$

One cycle of the IMM algorithm is as follows:

1. Model-conditioned reinitialization (for i =1,2,. . .,M):

Predicted mode probability: $\mu_{k|k-1}^{(i)} \triangleq P\left\{m_k^{(i)} \mid z^{k-1}\right\} = \sum_{j\in M} \pi_{ji}\mu_{k-1}^{(j)}$

Mixing weight: $\mu_{k-1}^{j|i} \triangleq P\left\{m_{k-1}^{(j)} \mid m_k^{(i)}, z^{k-1}\right\} = \dfrac{\pi_{ji}\mu_{k-1}^{(j)}}{\mu_{k|k-1}^{(i)}}$

Mixing estimate: $\bar{x}_{k-1|k-1}^{(i)} \triangleq E\left[x_{k-1} \mid m_k^{(i)}, z^{k-1}\right] = \sum_{j\in M} \hat{x}_{k-1|k-1}^{(j)}\mu_{k-1}^{j|i}$

Mixing covariance: $\bar{P}_{k-1|k-1}^{(i)} = \sum_{j\in M}\left[P_{k-1|k-1}^{(j)} + \left(\bar{x}_{k-1|k-1}^{(i)} - \hat{x}_{k-1|k-1}^{(j)}\right)\left(\bar{x}_{k-1|k-1}^{(i)} - \hat{x}_{k-1|k-1}^{(j)}\right)^T\right]\mu_{k-1}^{j|i}$

2. Model-conditioned filtering (for i =1,2,. . .,M):

Predicted state: $\hat{x}_{k|k-1}^{(i)} = F_{k-1}^{(i)}\bar{x}_{k-1|k-1}^{(i)} + G_{k-1}^{(i)}u_{k-1}^{(i)} + \Gamma_{k-1}^{(i)}\bar{w}_{k-1}^{(i)}$

Predicted measurement: $\hat{z}_{k|k-1}^{(i)} = H_k^{(i)}\hat{x}_{k|k-1}^{(i)} + \bar{v}_k^{(i)}$

Measurement residual: $\tilde{z}_k^{(i)} = z_k - \hat{z}_{k|k-1}^{(i)}$

Predicted covariance: $P_{k|k-1}^{(i)} = F_{k-1}^{(i)}\bar{P}_{k-1|k-1}^{(i)}\left(F_{k-1|k-1}^{(i)}\right)^T + G_{k-1}^{(i)}Q_{k-1}^{(i)}\left(G_{k-1}^{(i)}\right)^T$

Residual covariance: $S_k^{(i)} = H_k^{(i)}P_{k|k-1}^{(i)}\left(H_k^{(i)}\right)^T + R_k^{(i)}$

Filter gain: $K_k^{(i)} = P_{k|k-1}^{(i)}\left(H_k^{(i)}\right)^T\left(S_k^{(i)}\right)^{-1}$

Updated state: $\hat{x}_{k|k}^{(i)} = \hat{x}_{k|k-1}^{(i)} + K_k^{(i)}\tilde{z}_k^{(i)}$

Updated covariance: $P_{k|k}^{(i)} = P_{k|k-1}^{(i)} - K_k^{(i)}S_k^{(i)}\left(K_k^{(i)}\right)^T$

3. Mode probability update (for i =1,2,. . .,M):

Model likelihood: $L_k^{(i)} \triangleq p\left[\tilde{z}_k^{(i)} \mid m_{(i)}^k, z^{k-1}\right] = \dfrac{\exp\left[-\frac{1}{2}\left(\tilde{z}_k^{(i)}\right)'\left(S_k^{(i)}\right)^{-1}\tilde{z}_k^{(i)}\right]}{\left|2\pi S_k^{(i)}\right|^{1/2}}$

Mode probability: $\mu_k^{(i)} = \dfrac{\mu_{k|k-1}^{(i)}L_k^{(i)}}{\sum_{j\in M}\mu_{k|k-1}^{(j)}L_k^{(j)}}$

4. Estimate fusion:

Overall estimate: $\hat{x}_{k|k} = \sum_{i\in M}\hat{x}_{k|k}^{(i)}\mu_k^{(i)}$

Overall covariance: $P_{k|k} = \sum_{i\in M}\left[P_{k|k}^{(i)} + \left(\hat{x}_{k|k} - \hat{x}_{k|k}^{(i)}\right)\left(\hat{x}_{k|k} - \hat{x}_{k|k}^{(i)}\right)'\right]\mu_k^{(i)}$

One cycle of the GPB1 algorithm is as follows:

1. Model-conditioned reinitialization (for i =1,2,. . .,M):

Predicted mode probability: $\mu_{k|k-1}^{(i)} \triangleq P\left\{m_k^{(i)} \mid z^{k-1}\right\} = \sum_{j \in M} \pi_{ji} \mu_{k-1}^{(j)}$

2. Model-conditioned filtering (for i =1,2,. . .,M):

Predicted state: $\hat{x}_{k|k-1}^{(i)} = F_{k-1}^{(i)} \hat{x}_{k-1|k-1} + G_{k-1}^{(i)} u_{k-1}^{(i)} + \Gamma_{k-1}^{(i)} \overline{w}_{k-1}^{(i)}$

Predicted measurement: $\hat{z}_{k|k-1}^{(i)} = H_k^{(i)} \hat{x}_{k|k-1}^{(i)} + \overline{v}_k^{(i)}$

Measurement residual: $\tilde{z}_k^{(i)} = z_k - \hat{z}_{k|k-1}^{(i)}$

Predicted covariance: $P_{k|k-1}^{(i)} = F_{k-1}^{(i)} P_{k-1|k-1} \left(F_{k-1|k-1}^{(i)}\right)^T + G_{k-1}^{(i)} Q_{k-1}^{(i)} \left(G_{k-1}^{(i)}\right)^T$

Residual covariance: $S_k^{(i)} = H_k^{(i)} P_{k|k-1}^{(i)} \left(H_k^{(i)}\right)^T + R_k^{(i)}$

Filter gain: $K_k^{(i)} = P_{k|k-1}^{(i)} \left(H_k^{(i)}\right)^T \left(S_k^{(i)}\right)^{-1}$

Updated state: $\hat{x}_{k|k}^{(i)} = \hat{x}_{k|k-1}^{(i)} + K_k^{(i)} \tilde{z}_k^{(i)}$

Updated covariance: $P_{k|k}^{(i)} = P_{k|k-1}^{(i)} - K_k^{(i)} S_k^{(i)} \left(K_k^{(i)}\right)^T$

3. Mode probability update (for i =1,2,. . .,M):

Model likelihood: $L_k^{(i)} \triangleq p\left[\tilde{z}_k^{(i)} \mid m_{(i)}^k, z^{k-1}\right] = \dfrac{\exp\left[-\frac{1}{2}\left(\tilde{z}_k^{(i)}\right)' \left(S_k^{(i)}\right)^{-1} \tilde{z}_k^{(i)}\right]}{\left|2\pi S_k^{(i)}\right|^{1/2}}$

Mode probability: $\mu_k^{(i)} = \dfrac{\mu_{k|k-1}^{(i)} L_k^{(i)}}{\sum_{j \in M} \mu_{k|k-1}^{(j)} L_k^{(j)}}$

4. Estimate fusion:

Overall estimate: $\hat{x}_{k|k} = \sum_{i \in M} \hat{x}_{k|k}^{(i)} \mu_k^{(i)}$

Overall covariance: $P_{k|k} = \sum_{i \in M} \left[P_{k|k}^{(i)} + \left(\hat{x}_{k|k} - \hat{x}_{k|k}^{(i)}\right)\left(\hat{x}_{k|k} - \hat{x}_{k|k}^{(i)}\right)'\right] \mu_k^{(i)}$

One cycle of the GPB2 algorithm is as follows:

1. Model-conditioned filtering (for every transition (j,i)):

Predicted state: $\hat{x}_{k|k-1}^{(j,i)} = F_{k-1}^{(i)} \overline{x}_{k-1|k-1}^{(j)} + G_{k-1}^{(i)} u_{k-1}^{(i)} + \Gamma_{k-1}^{(i)} \overline{w}_{k-1}^{(i)}$

Predicted measurement: $\hat{z}_{k|k-1}^{(j,i)} = H_k^{(i)} \hat{x}_{k|k-1}^{(j,i)} + \overline{v}_k^{(i)}$

Measurement residual: $\tilde{z}_k^{(j,i)} = z_k - \hat{z}_{k|k-1}^{(j,i)}$

Predicted covariance: $P_{k|k-1}^{(j,i)} = F_{k-1}^{(i)} \overline{P}_{k-1|k-1}^{(j)} \left(F_{k-1|k-1}^{(i)}\right)^T + G_{k-1}^{(i)} Q_{k-1}^{(i)} \left(G_{k-1}^{(i)}\right)^T$

Residual covariance: $S_k^{(j,i)} = H_k^{(i)} P_{k|k-1}^{(j,i)} \left(H_k^{(i)}\right)^T + R_k^{(i)}$

Filter gain: $K_k^{(j,i)} = P_{k|k-1}^{(j,i)} \left(H_k^{(i)}\right)^T \left(S_k^{(j,i)}\right)^{-1}$

Updated state: $\hat{x}_{k|k}^{(j,i)} = \hat{x}_{k|k-1}^{(j,i)} + K_k^{(j,i)} \tilde{z}_k^{(j,i)}$

Updated covariance: $P_{k|k}^{(j,i)} = P_{k|k-1}^{(j,i)} - K_k^{(j,i)} S_k^{(j,i)} \left(K_k^{(j,i)}\right)^T$

2. Model-conditioned reinitialization (for every transition (j,i)):

Predicted mode probability: $\mu_{k|k-1}^{(i)} \triangleq P\left\{m_k^{(i)} \mid z^{k-1}\right\} = \sum_{j \in M} \pi_{ji} \mu_{k-1}^{(j)}$

Sequence mixing weight: $\mu_k^{j|i} \triangleq P\left\{m_{k-1}^{(j)} \mid m_k^{(i)}, z^{k-1}\right\} = \dfrac{\pi_{ji} \mu_{k-1}^{(j)}}{\mu_{k|k-1}^{(i)}}$

Mixing estimate: $\overline{x}_{k|k}^{(i)} \triangleq E\left[x_k \mid m_k^{(i)}, z^k\right] = \sum_{j \in M} \hat{x}_{k|k}^{(j,i)} \mu_k^{j|i}$

Mixing covariance: $\overline{P}_{k|k}^{(i)} = \sum_{j \in M} \left[P_{k|k}^{(j,i)} + \left(\overline{x}_{k|k}^{(i)} - \hat{x}_{k|k}^{(j,i)}\right)\left(\overline{x}_{k|k}^{(i)} - \hat{x}_{k|k}^{(j,i)}\right)^T\right] \mu_k^{j|i}$

3. Sequence probability update (for every transition (j,i)):

Sequence likelihood: $L_k^{(j,i)} \triangleq p\left[\tilde{z}_k^{(j,i)} \mid m_{(i)}^k, z^{k-1}\right] = \dfrac{\exp\left[-\frac{1}{2}\left(\tilde{z}_k^{(j,i)}\right)' \left(S_k^{(j,i)}\right)^{-1} \tilde{z}_k^{(j,i)}\right]}{\left|2\pi S_k^{(j,i)}\right|^{1/2}}$

Sequence probability: $\mu_k^{(j,i)} = \dfrac{\pi_{ji} \mu_{k-1}^{(j)} L_k^{(j,i)}}{\sum_{(j,i) \in M} \pi_{ji} \mu_{k-1}^{(j)} L_k^{(j,i)}}$

4. Estimate fusion:

Overall estimate: $\hat{x}_{k|k} = \sum_{(j,i) \in M} \hat{x}_{k|k}^{(j,i)} \mu_k^{(j,i)}$

Overall covariance: $P_{k|k} = \sum_{(j,i) \in M} \left[P_{k|k}^{(j,i)} + \left(\hat{x}_{k|k} - \hat{x}_{k|k}^{(j,i)}\right)\left(\hat{x}_{k|k} - \hat{x}_{k|k}^{(j,i)}\right)'\right] \mu_k^{(j,i)}$

One cycle of the B-Best algorithm is as follows:

1. Model-conditioned filtering for $\forall(j,i) \in \beta_k$

Predicted state: $\hat{x}_{k|k-1}^{(j,i)} = F_{k-1}^{(i)} \overline{x}_{k-1|k-1}^{(j)} + G_{k-1}^{(i)} u_{k-1}^{(i)} + \Gamma_{k-1}^{(i)} \overline{w}_{k-1}^{(i)}$

Predicted measurement: $\hat{z}_{k|k-1}^{(j,i)} = H_k^{(i)} \hat{x}_{k|k-1}^{(j,i)} + \overline{v}_k^{(i)}$

Measurement residual: $\tilde{z}_k^{(j,i)} = z_k - \hat{z}_{k|k-1}^{(j,i)}$

Predicted covariance: $P_{k|k-1}^{(j,i)} = F_{k-1}^{(i)} \overline{P}_{k-1|k-1}^{(j)} \left(F_{k-1|k-1}^{(i)}\right)^T + G_{k-1}^{(i)} Q_{k-1}^{(i)} \left(G_{k-1}^{(i)}\right)^T$

Residual covariance: $S_k^{(j,i)} = H_k^{(i)} P_{k|k-1}^{(j,i)} \left(H_k^{(i)}\right)^T + R_k^{(i)}$

Filter gain: $K_k^{(j,i)} = P_{k|k-1}^{(j,i)} \left(H_k^{(i)}\right)^T \left(S_k^{(j,i)}\right)^{-1}$

Updated state: $\hat{x}_{k|k}^{(j,i)} = \hat{x}_{k|k-1}^{(j,i)} + K_k^{(j,i)} \tilde{z}_k^{(j,i)}$

Updated covariance: $P_{k|k}^{(j,i)} = P_{k|k-1}^{(j,i)} - K_k^{(j,i)} S_k^{(j,i)} \left(K_k^{(j,i)}\right)^T$

2. Sequence probability update for $\forall(j,i) \in \beta_k$

Sequence likelihood: $L_k^{(j,i)} \triangleq p\left[\tilde{z}_k^{(j,i)} \mid m_{(i)}^k, z^{k-1}\right] = \dfrac{\exp\left[-\frac{1}{2}\left(\tilde{z}_k^{(j,i)}\right)' \left(S_k^{(j,i)}\right)^{-1} \tilde{z}_k^{(j,i)}\right]}{\left|2\pi S_k^{(j,i)}\right|^{1/2}}$

Sequence probability: $\mu_k^{(j,i)} = \dfrac{\pi_{ji} \mu_{k-1}^{(j)} L_k^{(j,i)}}{\sum_{(j,i) \in \beta_{k-1}} \pi_{ji} \mu_{k-1}^{(j)} L_k^{(j,i)}}$

3. Update the 'B' best sequences for time k

Most probable sequences β_k=B sequences that maximize $\mu_k^{(j,i)}$:

4. Sequence reinitialization for $\forall(j,i) \in \beta_k$

$$\overline{x}_{k|k}^{(i)} = \hat{x}_{k|k}^{(j,i)}$$
$$\overline{P}_{k|k}^{(i)} = P_{k|k}^{(j,i)}$$
$$\mu_k^{(i)} = \frac{\mu^{(j,i)}}{\sum_{(j,i) \in \beta} \mu^{(j,i)}}$$

5. Estimate fusion:

Overall estimate: $\hat{x}_{k|k} = \sum_{(j,i) \in \beta_k} \hat{x}_{k|k}^{(j,i)} \mu_k^{(j,i)}$

Overall covariance: $P_{k|k} = \sum_{(j,i) \in \beta_k} \left[P_{k|k}^{(j,i)} + \left(\hat{x}_{k|k} - \hat{x}_{k|k}^{(j,i)}\right)\left(\hat{x}_{k|k} - \hat{x}_{k|k}^{(j,i)}\right)'\right] \mu_k^{(j,i)}$

One cycle of the Viterbi algorithm is as follows:

1. Model-conditioned filtering (for every transition (j,i)):

Predicted state: $\hat{x}_{k|k-1}^{(j,i)} = F_{k-1}^{(i)} \overline{x}_{k-1|k-1}^{(j)} + G_{k-1}^{(i)} u_{k-1}^{(i)} + \Gamma_{k-1}^{(i)} \overline{w}_{k-1}^{(i)}$

Predicted measurement: $\hat{z}_{k|k-1}^{(j,i)} = H_k^{(i)} \hat{x}_{k|k-1}^{(j,i)} + \overline{v}_k^{(i)}$

Measurement residual: $\tilde{z}_k^{(j,i)} = z_k - \hat{z}_{k|k-1}^{(j,i)}$

Predicted covariance: $P_{k|k-1}^{(j,i)} = F_{k-1}^{(i)} \overline{P}_{k-1|k-1}^{(j)} \left(F_{k-1|k-1}^{(i)}\right)^T + G_{k-1}^{(i)} Q_{k-1}^{(i)} \left(G_{k-1}^{(i)}\right)^T$

Residual covariance: $S_k^{(j,i)} = H_k^{(i)} P_{k|k-1}^{(j,i)} \left(H_k^{(i)}\right)^T + R_k^{(i)}$

Filter gain: $K_k^{(j,i)} = P_{k|k-1}^{(j,i)} \left(H_k^{(i)}\right)^T \left(S_k^{(j,i)}\right)^{-1}$

Updated state: $\hat{x}_{k|k}^{(j,i)} = \hat{x}_{k|k-1}^{(j,i)} + K_k^{(j,i)} \tilde{z}_k^{(j,i)}$

Updated covariance: $P_{k|k}^{(j,i)} = P_{k|k-1}^{(j,i)} - K_k^{(j,i)} S_k^{(j,i)} \left(K_k^{(j,i)}\right)^T$

2. Sequence probability reinitialization (for every transition (j,i)):

Predicted mode probability: $\mu_{k|k-1}^{(i)} \triangleq P\left\{m_k^{(i)} \mid z^{k-1}\right\} = \sum_{j \in M} \pi_{ji} \mu_{k-1}^{(j)}$

Sequence mixing weight: $\mu_k^{j|i} \triangleq P\left\{m_{k-1}^{(j)} \mid m_k^{(i)}, z^{k-1}\right\} = \dfrac{\pi_{ji} \mu_{k-1}^{(j)}}{\mu_{k|k-1}^{(i)}}$

Sequence likelihood: $L_k^{(j,i)} \triangleq p\left[\tilde{z}_k^{(j,i)} \mid m_{(i)}^k, z^{k-1}\right] = \dfrac{\exp\left[-\dfrac{1}{2}\left(\tilde{z}_k^{(j,i)}\right)' \left(S_k^{(j,i)}\right)^{-1} \tilde{z}_k^{(j,i)}\right]}{\left|2\pi S_k^{(j,i)}\right|^{1/2}}$

Sequence probability: $\mu_k^{(j,i)} = \dfrac{\pi_{ji} \mu_{k-1}^{(j)} L_k^{(j,i)}}{\sum_{(j,i) \in M} \pi_{ji} \mu_{k-1}^{(j)} L_k^{(j,i)}}$

Most likely (j): $j = \arg\max_j \mu_k^{(j,i)}$

State reinitialization: $\overline{x}_{k|k}^{(i)} = \hat{x}_{k|k}^{(j,i)}$

Covariance reinitialization: $\overline{P}_{k|k}^{(i)} = \hat{P}_{k|k}^{(j,i)}$

Mode Probability $\mu_k^{(i)} = \dfrac{\mu_k^{(j,i)}}{\sum_j \mu_k^{(j)}}$

4. Estimate fusion:

Overall estimate: $\hat{x}_{k|k} = \sum_{i \in M} \overline{x}_{k|k}^{(i)} \mu_k^{(i)}$

Overall covariance: $P_{k|k} = \sum_{(i) \in M} \left[P_{k|k}^{(i)} + \left(\hat{x}_{k|k} - \overline{x}_{k|k}^{(i)}\right)\left(\hat{x}_{k|k} - \overline{x}_{k|k}^{(i)}\right)'\right] \mu_k^{(i)}$

One cycle of the RIMM algorithm is as follows:

1. Model-conditioned prediction probabilities (for i =1,2,. . .,M):

Predicted mode probability: $\mu_{k|k-1}^{(i)} \triangleq P\left\{m_k^{(i)} \mid z^{k-1}\right\} = \sum_{j \in M} \pi_{ji} \mu_{k-1}^{(j)}$

Mixing weight: $\mu_{k-1}^{j|i} \triangleq P\left\{m_{k-1}^{(j)} \mid m_k^{(i)}, z^{k-1}\right\} = \dfrac{\pi_{ji} \mu_{k-1}^{(j)}}{\mu_{k|k-1}^{(i)}}$

2. Model-conditioned prediction (for every transition (j,i)):

Mixing covariance: $P_{k|k-1}^{(j,i)} = \dfrac{\pi_{ji}}{\mu_{k|k-1}^{(i)}} F_k^{(i)} P_{k-1|k-1}^{(j)} \left(F_k^{(i)}\right)^T + G_k^{(i)} Q_k^{(i)} \left(G_k^{(i)}\right)^T$

Mixing estimate: $\hat{x}_{k|k-1}^{(j,i)} = E\left[x_k \mid z^{k-1}, m_{k-1}^{(j)}, m_k^{(i)}\right] = F_k^{(i)} \hat{x}_{k-1|k-1}^{(j)} \left(F_k^{(i)}\right)^T + G_k^{(i)} u_{k-1}^{(i)}$

Predicted state: $\hat{x}_{k|k-1}^{(i)} = P_{k|k-1}^{(i)} \sum_{j \in M} \left(P_{k|k-1}^{(j,i)}\right)^{-1} \hat{x}_{k|k-1}^{(j,i)} \mu_{k-1}^{(j|i)}$

Predicted covariance: $\left(P_{k|k-1}^{(i)}\right)^{-1} = \sum_{j \in M} \left(P_{k|k-1}^{(j,i)}\right)^{-1} \mu_{k-1}^{(j|i)}$

3. Model-conditioned filtering: (for i = 1,2,. . .M)

Predicted measurement: $\hat{z}_{k|k-1}^{(i)} = H_k^{(i)} \hat{x}_{k|k-1}^{(i)} + \bar{v}_k^{(i)}$

Measurement residual: $\tilde{z}_k^{(i)} = z_k - \hat{z}_{k|k-1}^{(i)}$

Predicted covariance: $P_{k|k-1}^{(i)} = F_{k-1}^{(i)} \bar{P}_{k-1|k-1}^{(i)} \left(F_{k-1|k-1}^{(i)}\right)^T + G_{k-1}^{(i)} Q_{k-1}^{(i)} \left(G_{k-1}^{(i)}\right)^T$

Residual covariance: $S_k^{(i)} = H_k^{(i)} P_{k|k-1}^{(i)} \left(H_k^{(i)}\right)^T + R_k^{(i)}$

Filter gain: $K_k^{(i)} = P_{k|k-1}^{(i)} \left(H_k^{(i)}\right)^T \left(S_k^{(i)}\right)^{-1}$

Updated state: $\hat{x}_{k|k}^{(i)} = \hat{x}_{k|k-1}^{(i)} + K_k^{(i)} \tilde{z}_k^{(i)}$

Updated covariance: $P_{k|k}^{(i)} = P_{k|k-1}^{(i)} - K_k^{(i)} S_k^{(i)} \left(K_k^{(i)}\right)^T$

4. Mode probability update (for i =1,2,. . .,M):

Model likelihood and Mode probability: **Same as IMM in the Appendix**

5. Estimate fusion:

Overall covariance: $P_{k|k}^{-1} = \sum_{i \in M} \left(P_{k|k}^{(i)}\right)^{-1} \mu_k^{(i)}$

Overall estimate: $\hat{x}_{k|k} = P_{k|k} \sum_{i \in M} \left(P_{k|k}^{(i)}\right)^{-1} \hat{x}_{k|k}^{(i)} \mu_k^{(i)}$

VITA

Ryan Pitre was born in Los Angeles, California in October 1977. He received a Bachelors of Science in electrical engineering from the University of New Orleans in 2002 and began working on his Master of Science in electrical engineering the following semester. He was awarded the Crescent City Doctoral Scholarship in his second semester of graduate school and plans continue his education and work towards a PhD in engineering and applied sciences at the University of New Orleans. His interests in the area of electrical engineering include target tracking, signal processing, and radar systems.